Analyse de grappe des données de catégories et de séquences : étude et application à la prédiction de la faillite personnelle

par

Tengke Xiong

thèse présentée au Département d'informatique
en vue de l'obtention du grade de philosophiae doctor (Ph.D.)

FACULTÉ DES SCIENCES
UNIVERSITÉ DE SHERBROOKE

Sherbrooke, Québec, Canada, juin 2011

Library and Archives Canada

Published Heritage Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et Archives Canada

Direction du Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 978-0-494-83344-5

Our file Notre référence
ISBN: 978-0-494-83344-5

CLUSTERING CATEGORICAL AND SEQUENCE DATA: INVESTIGATION AND APPLICATION IN PERSONAL BANKRUPTCY PREDICTION

Tengke Xiong
Faculté des Sciences
Université de Sherbrooke

A thesis submitted for the degree of
Doctor of Philosophy
June 2011

Le 23 juin 2011

le jury a accepté la thèse de Monsieur Tengke Xiong
dans sa version finale.

Membres du jury

Professeur Shengrui Wang
Directeur de recherche
Département d'informatique

Professeur André Mayers
Codirecteur de recherche
Département d'informatique

Professeur Ernest Monga
Évaluateur externe au programme
Département de mathématiques

Professeur Aijun An
Évaluateur externe
Department of Computer Science and Engineering
York University

Professeur Pierre-Marc Jodoin
Président rapporteur
Département d'informatique

Acknowledgements

First and foremost, I am grateful to my advisors Shengrui Wang and André Mayers. I have benefited so much in my research from the inspiration, motivation and encouragement that Prof. Wang constantly provided in the past four years. Prof. Mayers was always patient and gave me a lot of valuable advise and instruction. Working with them was really memorable and fun experience.

I also would like to thank Prof. Ernest Monga. Prof. Monga provided valuable suggestion on the project of personal bankruptcy prediction, I learned a lot from the discussion with him. Thanks are specially due to Vincent for his cooperation and help on the project.

I would like to thank my thesis committee members for giving me extremely valuable feedback on my thesis, their reviews and comments are really helpful to improve the thesis.

I would like to thank all my friends at UdeS, they made my years in Sherbrooke a fantastic experience.

Part of the research work presented in this thesis was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) Collaborative Research and Development (CRD) Grant. I was financially supported by the scholarship obtained from the NSERC CRD project and research funds of Prof. Wang during my studies at the Université de Sherbrooke. I would like to express my sincere thanks to NSERC.

Finally, I would like to thank my family for their unconditional and untiring support and contribution. I could not finish my degree without my family. Their love is my worthless treasure.

Sommaire

L'analyse de grappes est l'une des techniques les plus importantes utilisées dans le forage de données. Elle a de nombreuses applications dans l'extraction de motifs, la recherche d'information, la synthèse d'information, la compression, etc. Les travaux de recherche de cette thèse portent sur le regroupement des données catégoriques et séquentielles. Il s'agit d'une tâche beaucoup plus difficile que le regroupement des données numériques, due au manque de mesure de similarité évidente entre les données catégoriques et entre les séquences catégoriques. Dans cette thèse, nous avons conçu des algorithmes efficaces pour regrouper des données et des séquences catégoriques. Nos études expérimentales permettent de démontrer les performances supérieures de nos algorithmes par rapport aux algorithmes existants. Nous avons aussi appliqué des algorithmes proposés pour résoudre le probème de prédiction de la faillite personnelle.

Le regroupement des données catégoriques pose deux défis : définir une mesure de similarité significative, et traiter efficacement les groupes qui résident dans des sous-espaces différents. Dans cette thèse, nous considérons l'analyse de grappes dans une perspective d'optimisation et proposons une nouvelle fonction objectif. En se basant sur cette formulation, nous concevons un nouvel algorithme hiérarchique par division, nommé DHCC, pour les données catégoriques. Dans la procédure de bisection du DHCC, l'initialisation de la division est basée sur l'analyse des correspondances multiples (ACM). Nous élaborons une stratégie pour pallier à un problème clé de l'approche par division, à savoir quand il n'est plus nécessaire de diviser. L'algorithme proposé est entièrement automatique (sans paramètre, aucune hypothèse concernant le nombre de groupes), indépendant de l'ordre dans lequel les données sont traitées,

extensible à de grands ensembles de données, et finalement. capable de découvrir naturellement des groupes inclus dans des sous-espaces.

La connaissance a priori sur les données peut être incorporée dans le processus de l'analyse de grappes, ce qui est connu sous de nom de l'analyse de grappes semi-supervisée, pour procurer une amélioration considérable pour la qualité de l'analyse de grappes. Dans cette thèse, nous considérons l'analyse de grappes semi-supervisée comme un problème d'optimisation avec contraintes au niveau des instances et proposons une approche automatique pour guider le processus de l'optimisation sous contraintes. Ceci nous permet de proposer un nouvel algorithme semi-supervisé hiérarchique par division pour les données catégoriques, nommé SDHCC. Notre algorithme ne nécessite pas la fixation de paramètre, ne comporte aucune opération sensible à l'ordre de traitement de données et est efficace en prenant l'avantage des connaissances au niveau de contraintes d'appartenance des instances pour améliorer la qualité des résultats.

De nombreux algorithmes de regroupement des séquences s'appuient sur une mesure de similitude entre des paires de séquences. Habituellement, une telle mesure est efficace s'il y a beaucoup d'informations dans les motifs retrouvés parmi ces séquences. Toutefois, il est difficile de définir une mesure de similarité significative pour les paires de séquences si celles-ci sont courtes et contiennent du bruit. Dans cette thèse, nous contournons cet obstacle en définissant une mesure de similitude entre une séquence individuelle et un ensemble de séquences en se basant sur un modèle de la distribution de probabilité conditionnelle. À partir de cette mesure, nous concevons un nouvel algorithme K-moyennes basé sur le modèle pour l'analyse de grappes de séquences. Cet algorithme fonctionne de façon similaire au traditionnel algorithme K-moyennes pour les données vectorielles.

Enfin, nous avons développé un système pour la prédiction de faillites personnelles dont les attributs de prédictions sont principalement les attributs de faillites découverts par les techniques de regroupement proposés dans cette thèse. Les caractéristiques de faillites découvertes sont représentées dans un espace vectoriel de basses dimensions. À partir du nouvel espace d'attributs, qui peut être complété avec des attributs existants de prédiction connus (par exemple, le score de crédit),

un classificateur basé sur la machine à vecteur de support (SVM) est développé pour combiner ces différents attributs. Les résultats expérimentaux démontrent que notre système est prometteur pour la prédiction au niveau de la performance et au niveau de l'explication qu'il peut fournir.

Abstract

Cluster analysis is one of the most important and useful data mining techniques, and there are many applications of cluster analysis in pattern extraction, information retrieval, summarization, compression and other areas. The focus of this thesis is on clustering categorical and sequence data. Clustering categorical and sequence data is much more challenging than clustering numeric data because there is no inherently meaningful measure of similarity between the categorical objects and sequences. In this thesis, we design novel efficient and effective clustering algorithms for clustering categorical data and sequence respectively, and we perform extensive experiments to demonstrate the superior performance of our proposed algorithm. We also explore the extent to which the use of the proposed clustering algorithms can help to solve the personal bankruptcy prediction problem.

Clustering categorical data poses two challenges: defining an inherently meaningful similarity measure, and effectively dealing with clusters which are often embedded in different subspaces. In this thesis, we view the task of clustering categorical data from an optimization perspective and propose a novel objective function. Based on the new formulation, we design a divisive hierarchical clustering algorithm for categorical data, named DHCC. In the bisection procedure of DHCC, the initialization of the splitting is based on multiple correspondence analysis (MCA). We devise a strategy for dealing with the key issue in the divisive approach, namely, when to terminate the splitting process. The proposed algorithm is parameter-free, independent of the order in which the data is processed, scalable to large data sets and capable of seamlessly discovering clusters embedded in subspaces.

The prior knowledge about the data can be incorporated into the clustering process, which is known as semi-supervised clustering, to produce considerable improvement in learning accuracy. In this thesis, we view semi-supervised clustering of categorical data as an optimization problem with extra instance-level constraints, and propose a systematic and fully automated approach to guide the optimization process to a better solution in terms of satisfying the constraints, which would also be beneficial to the unconstrained objects. The proposed semi-supervised divisive hierarchical clustering algorithm for categorical data, named SDHCC, is parameter-free, fully automatic and effective in taking advantage of instance-level constraint background knowledge to improve the quality of the resultant dendrogram.

Many existing sequence clustering algorithms rely on a pair-wise measure of similarity between sequences. Usually, such a measure is effective if there are significantly informative patterns in the sequences. However, it is difficult to define a meaningful pair-wise similarity measure if sequences are short and contain noise. In this thesis, we circumvent the obstacle of defining the pairwise similarity by defining the similarity between an individual sequence and a set of sequences. Based on the new similarity measure, which is based on the conditional probability distribution (CPD) model, we design a novel model-based K-means clustering algorithm for sequence clustering, which works in a similar way to the traditional K-means on vectorial data.

Finally, we develop a personal bankruptcy prediction system whose predictors are mainly the bankruptcy features discovered by the clustering techniques proposed in this thesis. The mined bankruptcy features are represented in low-dimensional vector space. From the new feature space, which can be extended with some existing prediction-capable features (e.g., credit score), a support vector machine (SVM) classifier is built to combine these mined and already existing features. Our system is readily comprehensible and demonstrates promising prediction performance.

Contents

List of Figures

Chapter 1

Introduction

Cluster analysis is one of the most important and useful data mining techniques [32, 37, 46, 77]. Clustering is an exploratory learning process whose aim is to group unlabeled data into meaningful clusters, so that objects in the same cluster show great similarity and objects from distinct clusters show great dissimilarity. The concept of similarity can be defined in many different ways, according to the purpose of the analysis, domain-specific assumption and prior background knowledge about the data. There are many applications of cluster analysis in pattern extraction, information retrieval, summarization, compression and other areas. Most of the clustering algorithms present in the literature focus on data sets where the objects are defined on a set of numerical values. In such a case, the similarity of the objects can be decided using well-studied measures derived from geometric analogies, such as Euclidean distance. While categorical data is commonly seen in many real-life applications, where the elements of the data are non-numeric and nominal, imposing more challenge for cluster analysis in these domains.

The investigation of clustering categorical and sequence data arises from our project on a major Canadian bank. In our project, original high-dimensional complex data are transformed into categorical data, which mainly takes two forms: categorical tuples and sequences. We resort to clustering techniques to discover the comprehensible features that can distinguish bad accounts from good ones. The focus of this thesis is on clustering categorical and sequence data, and the application of the proposed

clustering techniques in personal bankruptcy prediction. In this chapter, we will give an overview of traditional unsupervised and semi-supervised learning, and then describe the clustering method for categorical and sequence data, and then present an introduction to personal bankruptcy prediction. We conclude this chapter with a discussion of the thesis contributions.

1.1 Unsupervised learning

Two of the most important tasks in the field of data mining are classification and clustering [30, 85]. Classification is a supervised learning technique, whereas clustering is completely unsupervised. The aim of clustering is to group a set of objects into clusters without the guidance of prior background knowledge. Clustering techniques are broadly divided into hierarchical and partitioning, depending on whether the algorithm generates a hierarchical clustering structure or a flat partition of the data set.

1.1.1 Hierarchical clustering

In hierarchical clustering, the objects are placed in different clusters, and these clusters have ancestor-descendant relationship. Usually, a binary tree is employed to represent the dendrogram structure of the clustering results. The dendrogram can be cut at different levels to generate different clustering of the data. Hierarchical clustering algorithms can be further divided into agglomerative (bottom-up) and divisive (top-down) methods:

- Agglomerative methods: Start with each object as a singleton cluster and, at each step, merge the closest pair of clusters according to the similarity measure. The most commonly used methods to measure the similarity between pairwise clusters are single-link, complete-link, and group average. Agglomerative methods suffer from high time complexity and the problem that early wrong decision of merging two small clusters can expand to following merges, leading to undesirable final clustering tree structure. Fisher (1996) studied iterative hierarchical

cluster redistribution to improve once constructed dendrogram [33]. Karypis et al. (1999) also researched refinement for hierarchical clustering [48]. However, the global refinement procedure destroys the desirable hierarchical clustering structure [15, 90]. A basic agglomerative hierarchical clustering algorithm is illustrated in Figure 1.1.

- Divisive methods: Start with one, all-inclusive cluster containing all the objects and, at each step, split a cluster until only singleton clusters of individual objects remain or the termination criterion is satisfied. The two most important issues of divisive methods are how to split a cluster and how to choose the next cluster to split or when to terminate splitting (if not intending to generate the whole clustering tree).

Algorithm: Agglomerative Hierarchical Clustering

1. Place each object in its own cluster. Computer the proximity matrix containing the distance between each pair of objects.
2. Repeat until the number of clusters reaches one.
   (a) Find the most similar pair of clusters C_i and C_j using the proximity matrix. Merge clusters C_i and C_j to a new cluster C_p.
   (b) Remove C_i and C_j from current clusters, and update the proximity matrix by adding the distance between cluster C_p and other clusters.

Figure 1.1: Agglomerative hierarchical clustering algorithm

1.1.2 Partitioning clustering

In partitioning clustering, given a data set and the number of clusters K, an algorithm divides data into K subsets, in which, each subset represents a cluster. Partitioning

clustering is further divided into hard clustering and fuzzy clustering. In hard clustering, each object belongs to exactly one cluster. In fuzzy clustering, each object is allowed to belong to two or more clusters, associated with a set of membership degrees [43]. In this thesis, we only consider assigning each object to exactly one cluster.

Partitioning clustering exploits iterative relocation to optimize the partition of the data set. Unlike traditional hierarchical methods, in which clusters are not revisited after being constructed, an algorithm of partitioning clustering tries to discover the clusters by iteratively reassigning objects between the K clusters.

The number of clusters K is assumed as a prior known parameter in most partitioning methods. Learning the 'true' number of clusters in a given data set is a fundamental and largely unsolved problem [75, 76]. In hierarchical clustering, this problem is less critical, as the hierarchical clustering structure offers more flexibility to analyzing the data at different levels of similarity. A partition of the data in hierarchical clustering can be obtained by cutting the clustering tree at certain level.

There are a number of techniques in partitioning clustering, we list some as follows. In the center-based method, the most representative object within a cluster is selected to represent the cluster (K-medoids [64]), or each cluster is represented by the mean of its objects, which is call centroid (K-means). In the density-based method, a cluster is defined as a connected dense component against surrounding region, a representative algorithm is DBSCAN [31]. In the probabilistic method, each object is considered to be a sample independently drawn from a *mixture model* of several probability distributions, and the *Expectation-Maximization* (EM) technique is exploited to optimize the overall likelihood of generating the whole data set [73]. In the graph-theoretic method, the clustering problem is modeled by a graph $G = (V, E)$, where each vertex $v_i \in V$ corresponds to a data object, and each edge $e_{ij} \in E$ corresponds to the similarity between data objects x_i and x_j according to a domain-specific measure, and discovering the K clusters is equivalent to finding the K *minimum cut* (MC). the definition of the cut of a graph can be found in [36].

In partitioning clustering, the K-means [41] is by far the most popular clustering tool used in scientific and industrial application [85]. A basic K-means algorithm is

illustrated in Figure 1.2.

Algorithm: Basic K-means

1. (Randomly) select K objects as the initial centroids.
2. Assign all objects to the closest centroid.
3. Recalculate the centroid of each cluster.
4. Repeat steps 2 and 3 until the centoids don't change.

Figure 1.2: Basic K-means algorithm

1.2 Semi-supervised learning

Recently, there have been great interests in investigation of the correlation between completely supervised and unsupervised learning [16, 65], resulting in the rise of two research branches: semi-supervised classification, where the unlabeled data is used in the learning process to improve classification accuracy; and semi-supervised clustering, where partially labeled data or pairwise constraints is used to aid unsupervised clustering. A good review of semi-supervised learning methods is given in [96]. In this thesis, we focus on semi-supervised clustering of categorical data.

Compared with unsupervised learning, semi-supervised learning is a class of machine learning techniques that make use of both labeled and unlabeled data for training [16]. In semi-supervised clustering, prior knowledge is incorporated into the clustering process to produce considerable improvement in learning accuracy. In real applications, some background information about the data may exist, such as a small number of labeled instances, or pairwise instance-level constraints indicating that two instances should (*must-link*) or should not (*cannot-link*) be associated with the same cluster. How to take advantage of this background knowledge in cluster analysis is a subject of growing interest for the data mining community.

Existing methods for semi-supervised clustering can be generally grouped into two categories: the prior knowledge is incorporated into the clustering process either by

modifying the search for appropriate clusters or by adapting the similarity measure (or distortion called in some literature).

- In *search-based* methods. the clustering algorithm itself is modified so that the available labels or constraints can be used to bias the search for an appropriate clustering. This can be done in several ways, such as by enforcing constraints to be satisfied during cluster assignment in the clustering process [23, 82], by initializing the clusters from the transitive closures obtained from available labels or constraints [8], by projecting original data space to a low-dimensional space, where the projection matrix is obtained from optimization of the objective function reflecting the satisfaction of constraint knowledge [78] or by modifying the clustering objective function so that it includes a penalty for constraint violation [9] or a reward for constraint satisfaction [54].

- In *similarity-adapting* methods, the similarity measure used in unsupervised algorithm is adapted, so that the available constraints can be more easily satisfied. Several similarity measures, or distortion measures named in some literature, have been used for similarity-adapting semi-supervised clustering. For example, string-edit distance trained using EM [12], parameterized Euclidean or Mahalanobis distances trained using convex optimization [6, 11, 87], Euclidean distance modified by shortest-path algorithm [51].

1.3 Clustering categorical and sequence data

Clustering categorical and sequence data is much more challenging than clustering numeric data because there is no inherently meaningful measure of similarity between the categorical objects or sequences.

1.3.1 Clustering categorical data

Clustering categorical data is an important task. Categorical data is commonly seen in many fields, including the social and behavioral sciences, statistics, psychology, etc.

One special type of categorical data is transactional data, where the term transaction refers to a collection of items generally covering many domains: for example, the market basket of a consumer or the set of symptoms presented by a patient. With large amounts of categorical data being generated in real-life applications, clustering categorical data has been receiving increasing attention in recent years

Clustering categorical data poses two challenges: defining an inherently meaningful similarity measure, and effectively dealing with clusters which are often embedded in different subspaces. The detailed explanation is as follows.

Due to the lack of inherently meaningful measure of the similarity between categorical objects, various similarity or distance measures have therefore been proposed in recent years for clustering categorical and transactional data. While some pairwise similarity measures, such as the cosine measure, the Dice and Jaccard coefficients, etc. can be used for the comparison of categorical data [77], it is commonly believed that a pairwise similarity measure is not suitable for this purpose [40, 83, 92]. For sets X and Y of items, the *Dice* coefficient is defined as:

$$Dice = \frac{2\,|X \cap Y|}{|X| + |Y|}$$

The *Jaccard* coefficient is defined as:

$$Jaccard = \frac{|X \cap Y|}{|X \cup Y|}$$

For example, consider a set of five transactions, t_1=$\{a,\ b,\ d,\ f\}$, t_2=$\{b,\ e,\ g\}$, t_3=$\{a,\ c,\ h,\ i\}$, t_4=$\{a,\ b,\ c\}$, t_5=$\{b,\ c,\ j,\ k\}$, where t_i denotes a transaction consisting of a set of items corresponding to the categories of merchandise or service involved in the transaction. We can see that some pairs of transactions share few items: for instance, t_2 and t_3 share no items, while t_2 and t_4 share only one. In this case, the *Jaccard* coefficient between t_2 and t_3 is 0, and that between t_2 and t_4 is 0.2, thus these transactions cannot be grouped together using the notion of pairwise similarity. However, viewed globally, the items a, b, c are frequent items in these transactions and this could serve as a major characteristic on which to group these transactions

together.

Another issue in clustering categorical data is how to effectively deal with clusters which have a greater tendency to be embedded in different, possibly overlapping, subspaces. For example, in grouping customers based on transactional data, different groups of customers are distinguished by different purchasing habits, while customers in the same group have a similar interest in certain items. Also, in the social and behavioral sciences, different groups of people exhibit different social habits and behaviors. Unlike conventional numeric data, the domains of the attributes in categorical data are discrete and small: for example, the binary attribute with values 'yes' and 'no' is commonly seen in categorical data. Therefore, clusters in these data are distinguished by differences in the subspaces in which they are formed. Traditional clustering algorithms, which search for clusters defined on the whole dimension, have difficulty in discovering these clusters formed in subspaces, especially when the dimensionality of the subspaces is small.

1.3.2 Clustering sequence data

In the past few years, we have seen a rapid increase in the amount of sequence data. The sequence data are commonly seen in many scientific and business domains, such as genomic DNA sequences, unfolded protein sequences, text documents, web usage data, behavior or event sequences etc. The analysis of sequence data becomes an interesting and important research area because there is an increasing need to develop methods to analyze large amounts of sequence data efficiently.

A number of approaches have been investigated in the domain of sequence clustering. The clustering results can potentially reveal unknown object categories that lead to a better understanding of the nature of the sequence, for example, discover the unknown functions of a protein sequence. The nearest neighbor technique based on *edit distance* is one of the preferred methods for sequence clustering [2, 24, 63, 93]. Given two sequences s_1 and s_2, the edit distance between them is minimum number of edit operations required to transform s_1 into s_2. Most commonly, the allowable edit operations are insertion, deletion, or substitution of a single character. For these

operations, *edit distance* is also called *Levenshtein* distance. Many existing sequence clustering algorithms rely on such pairwise measure of similarity between sequences. Usually, such a measure is effective if there are significantly informative patterns in the sequences. However, it is difficult to define a meaningful pairwise similarity measure if sequences are short and contain noise [88].

1.4 Personal bankruptcy prediction

Personal bankruptcy prediction has been of increasing concern in both the industry and academic community, as bankruptcy results in significant losses to creditors. In credit card portfolio management, bankruptcy prediction is a key measure to prevent the accelerating losses resulting from personal bankruptcy. There were 90,610 personal bankruptcy cases (excluding proposals) in Canada in 2008 [1], more than four times the figure for 1988. The total personal bankruptcy debt in 2008 was $7.414 billion, whereas it was less than $1 billion in 1987. It is also reported in *Industry Canada* that 87.4% of personal bankruptcy cases involved credit card debt, which is the most frequently reported type of debt. To address this problem, besides carefully evaluating the creditworthiness of credit card applicants at the very beginning, credit card issuers must make a greater effort to identify potential bad accounts whose owners will go bankrupt over the life of the credit, because many clients whose creditworthiness was good when they applied for credit ultimately went bankrupt. From the creditor's standpoint, the earlier bad accounts are identified, the lower the losses entailed, which can be seen in Figure 1.3. The figure is computed from our project data (Master credit card data from one major Canadian bank), which shows the relationship between the debt of bankrupt accounts and the period before going bankrupt. We can see that the debt of bankrupt accounts increases linearly when approaching bankruptcy. However, early identification represents a greater challenge, which will be illustrated in Chapter 5.

In our investigation, we aim to design a prediction system running on a credit card data base, which is extensible, i.e., able to integrate existing prediction-capable

[1]Industry Canada. Available: `http://www.ic.gc.ca/ic_wp-pa.htm`

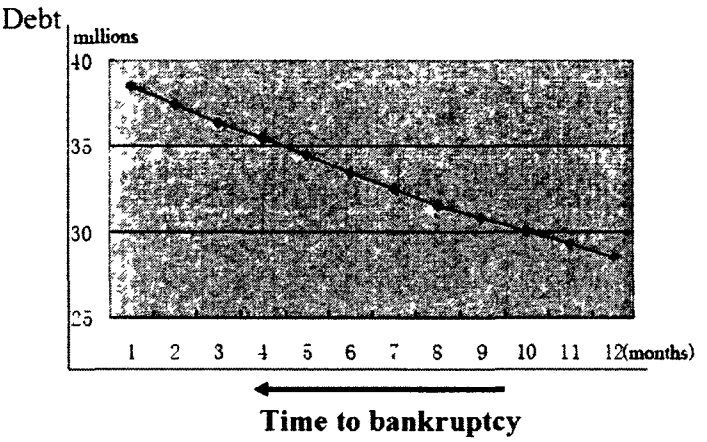

Figure 1.3: The relationship between debt and period before bankruptcy

features, either from data mining or domain expertise (e.g., credit scores); it is also readily comprehensible and can be used in industrial applications. The original purpose of our investigation was to complement existing prediction models, especially the credit scoring models, by identifying the bad accounts they tended to miss.

1.5 Thesis contributions

The contributions of this thesis are outlined below:

- We formulize the task of clustering categorical data from an optimization perspective, and set the objective to optimize the objective function, which is the sum of Chi-square error (SCE). Starting from this, we present the mathematical derivation of the definition of cluster center for categorical data. For details, see Chapter 2.

- We design a simple and systematic, yet efficient and effective divisive hierarchical clustering algorithm for categorical data, called DHCC, which is parameter-free, order-independent, and scalable to large data sets. We exploit a new data presentation for categorical data, based on which we employ the Chi-square statistic in a novel manner in dissimilarity calculation, making DHCC capable

of seamlessly discovering clusters embedded in subspaces. The detailed design of DHCC is presented in Chapter 2.

- We also view semi-supervised clustering of categorical data as a problem of optimizing our defined objective function (SCE) subject to extra constraint, and propose a systematic approach to deal with this problem. A novel semi-supervised divisive hierarchical algorithm for clustering categorical data, named SDHCC is described in Chapter 3.

- We propose a statistical model of sequence similarity. It is robust to noise and suitable for categorical sequences. Based on the model, a novel model-based K-means algorithm is designed for clustering categorical sequences and can be adapted to ordinal sequences. The statistical model and the model-based K-means are described in Chapter 4.

- We design and implement a personal bankruptcy prediction system running on a credit card data base. The system is extensible, being able to combine the knowledge discovered by data mining and domain expertise. The bankruptcy features are discovered by using our proposed techniques mentioned above. The detailed implementation of the prediction is presented in Chapter 5.

Apart from the chapters mentioned above, Chapter 6 concludes the thesis and presents the directions for future research.

Chapter 2

Clustering Categorical Data

This chapter views the task of clustering categorical data from an optimization perspective and describes a novel divisive hierarchical clustering algorithm for categorical data, named DHCC [89, 90]. We propose effective procedures to initialize and refine the splitting of clusters. The initialization of the splitting is based on multiple correspondence analysis (MCA). We also devise a strategy for deciding when to terminate the splitting process. The proposed algorithm has five merits. First, due to its hierarchical nature, our algorithm yields a dendrogram representing nested groupings of patterns and similarity levels at different granularities. Second, it is parameter-free, fully automatic and, in particular, requires no assumption regarding the number of clusters. Third, it is independent of the order in which the data is processed. Fourth, it is scalable to large data sets. And finally, our algorithm is capable of seamlessly discovering clusters embedded in subspaces thanks to its use of a novel data representation and Chi-square dissimilarity measures.

2.1 Introduction

The DHCC algorithm is based on multiple correspondence analysis (MCA), a powerful factor analysis tool for categorical data which is widely used in the social and behavioral sciences [1, 38, 39]. MCA has been employed in on-line analytical processing (OLAP) to reorganize a query result presented in the form of a data cube,

in order to enhance visual representation of the cube [62]. This work inspired us to design an efficient and effective algorithm for clustering categorical data based on MCA. In DHCC, MCA plays an important role in analyzing the data globally to perform initial bisection. To the best of our knowledge, DHCC is the first divisive hierarchical algorithm for clustering categorical data [89].

A hierarchical clustering algorithm yields a dendrogram representing nested groupings of patterns and similarity levels at different granularities, which offers more flexibility for exploratory analysis, and some studies suggest that hierarchical algorithms can produce better-quality clusters [46, 77]. Compared with agglomerative approaches, divisive algorithms have received less investigation. However, recent research suggests that divisive algorithms outperform agglomerative algorithms in terms of computational complexity and cluster quality [26, 95]. The divisive approach in hierarchical clustering is superior to the local computing-based agglomerative approach because it allows global information on the data distribution to be taken into account in detecting clusters.

A nice characteristic of DHCC is that it can discover clusters embedded in subspaces. In DHCC, the original categorical data set is represented in a Boolean vector space, where each categorical value represents a dimension. Like the iterative top-down subspace clustering algorithm for numeric data [3, 13, 66], where the individual attributes are weighted differently in each cluster to determine the subspace forming the cluster, the similarity measure in our algorithm also treats the dimensions differently in each cluster, according to their association with the cluster. However, it does not explicitly involve attribute-weight calculation to determine the subspace associated with each cluster, as do certain algorithms for numeric data [13] and categorical data [34]. Thus, DHCC is capable of seamlessly discovering clusters embedded in subspaces of the original data space.

2.2 Related work

In this section, we present and discuss existing methods for clustering categorical data. With the upsurge in the amount of categorical data in many fields, the problem of

automatically clustering large amounts of categorical data has become increasingly important and has been widely investigated recently [5, 7, 15, 18, 27, 34, 35, 40, 45, 57, 72, 83, 92, 94]. However, each of the existing approaches suffers from one or more of the following drawbacks, which have only been addressed efficiently for numerical data clustering [32, 46, 50, 77]:

- The need to set input parameters, such as an assumed number of clusters. Parameter-laden algorithms present several problems [50, 77]. It can be difficult to tune the parameters, and even more challenging if a small change in the parameters drastically changes the clustering results. This in turn makes the use of such a method tricky in practical applications. Additionally, Keogh et al. (2004) have established empirically that in the context of an anomalous situation, parameters tuned to fit one data set completely fail to fit a new but similar data set. Hence the conventional wisdom is that, for clustering algorithms, "the fewer parameters, the better, ideally none"[50, 77].

- Dependence on the order in which the data is processed. For algorithms subject to this drawback, an object may be mistakenly assigned to a wrong cluster because some prior objects have been 'inappropriately' processed. The COOLCAT algorithm [7] is an example of this. It is unreasonable for an algorithm to output different – even drastically different – clustering results for the same data set presented in a different processing order. A good algorithm should thus be order-independent.

- High complexity, preventing some algorithms from being used on very large data sets. The time complexity of some algorithms is quadratic with respect to the number of objects n; the ROCK algorithm [40] is a case in point. To solve the high-complexity problem, a sampling technique is employed as the initialization step. First, the algorithm is run on the sample objects, which have a much smaller scope in terms of quantity, and then the rest of objects are assigned to the clusters generated from the sample objects. The quality of clustering depends heavily on the samples, making the results unstable. For example, if no object from a true cluster is sampled, then that cluster cannot be

generated and all the objects from the cluster will be assigned inappropriately. So a good algorithm should be scalable to large data sets.

Several existing mainstream algorithms used to cluster categorical data are presented and discussed as follows.

The ROCK algorithm presented in [40] extends the Jaccard coefficient similarity measure by exploiting the concept of *neighborhood*: i.e., a pair of points X_i and X_j are neighbors if $sim(X_i, X_j) \geq \theta$, where the function sim is the Jaccard coefficient. The similarity between X_i and X_j is calculated based on *links*, i.e., the number of neighbors X_i and X_j have in common. ROCK is an agglomerative hierarchical clustering algorithm based on the extension of the pairwise similarity measure. Its clustering performance, however, depends heavily on the threshold θ, and it is difficult to make the right choice of θ in practical applications. The time complexity of ROCK is $O(n^2 + nm_m m_a + n^2 \log n)$, where m_m is the maximum number of neighbors, m_a is the average number of neighbors and n is the number of objects for clustering. The high complexity prevents the use of this algorithm on very large data sets.

Instead of using pairwise similarity measures, some extensions of the traditional K-means algorithm, such as the K-modes algorithms, seek to measure the similarity between an individual categorical object and a set of objects [27, 45, 72]. The definition of similarity between an individual categorical object and a set of categorical objects is more meaningful, especially when the clusters are well established. In the K-modes algorithms, the mean of a cluster is replaced by the mode to represent the cluster, and the distance between an object and a model is redefined. For example, in [72], a mode of a cluster is represented by $Q = (q_1, \cdots, q_m)$ with $q_j = \{(v_j, f_{v_j}) | v_j \in D_j\}$, where v_j is a categorical value of attribute A_j, whose domain is D_j, and f_{v_j} is the relative frequency of category v_j within the cluster. The distance between an object X and a cluster whose mode is Q is defined by $d(X, Q) = \sum_{j=1}^{m} (1 - f_{x_j})$, where f_{x_j} is the relative frequency of category x_j in the cluster. The performance of the K-modes algorithms relies heavily on the initialization of the K modes.

The CACTUS algorithm [35] defines a cluster as a subset of the Cartesian product of the domains of all the attributes. Candidate clusters are expanded from inter-attribute summaries and intra-attribute summaries. The final clusters are determined

by deleting false candidates, i.e., those with little support. The support threshold is set to α times the expected support of the cluster under the attribute independence assumption, where α is an important parameter which is difficult to tune. Additionally, this formalized definition of a cluster is based on the hypothesis that the clusters are formed over the entire original data space. However, this is impractical for real-life data sets, as clusters are more likely to form in different subspaces in categorical data. Therefore, CACTUS may fail in generating clusters in practice.

Some approaches apply information-theory concepts such as entropy in algorithm design [7, 57]. The goal of these approaches is to seek an optimum grouping of the objects such that the entropy is the smallest. The COOLCAT algorithm [7] employs the notion of entropy in assigning unclustered objects. Given an initial set of clusters, assignment of X_i is done by choosing the cluster such that the entropy of the resulting clustering is minimum. The incremental assignment finishes when every object has been placed in some cluster. The order in which the objects are processed has a definite impact on the clustering quality.

The agglomerative hierarchical clustering algorithm LIMBO [5] uses the information bottleneck method to build a Distributional Cluster Feature (DCF) tree. In this process, a preliminary clustering is done and the statistical features are stored in the leaf nodes of the DCF tree. The leaf nodes are then further clustered, using an agglomerative hierarchical approach. The generation of the DCF tree is affected by three parameters: the branching factor B, the maximum space bound S and the maximum DCF entry size E.

Subspace clustering for categorical data has been studied in recent years. SUBCAD [34] is designed for clustering high-dimensional categorical data. The algorithm exploits an objective function which combines compactness and separation measures for both object relocation and subspace determination. SUBCAD consists of two phases, initialization and optimization. In the initialization phase, a sampling technique is used to generate an initial grouping. In the optimization phase, relocation is carried out to minimize the objective function, and if an object is relocated, the two related clusters are updated immediately, including the occurrence numbers of the categorical values and the associated subspaces. This incremental relocation may

lead the clustering process to evolve differently for different processing orders. and may ultimately result in different clustering results

Parameter-free approaches for clustering categorical data have had great appeal. Cesario et al. (2007) propose a top-down parameter-free algorithm, AT-DC, for clustering categorical data. The algorithm consists of two procedures, splitting and stabilization. Based on the proposed clustering quality measure, the goal of both procedures is to yield improvement in the quality of the partition. The splitting of a cluster C_p begins with two initial subclusters, one empty and the other containing all the objects of C_p, and then iteratively relocates the objects in C_p to improve the quality according to the defined measure. The splitting procedure is followed by a global refinement process like the K-means; the iterative refinement is called the stabilization procedure. The algorithm terminates when no further improvement can be achieved. The global refinement process destroys the hierarchical structure, which makes AT-DC a partitional rather than a hierarchical algorithm. The algorithm has the great appeal of being parameter-free; however, the processing order in both splitting and stabilization has impact on the clustering quality, which constitutes a major drawback.

The drawbacks of these approaches are summarized in Table 2.1. It is worth noting here that the algorithms with quadratic time complexity which use a sampling technique in the original papers are considered non-scalable, because we are only concerned with the inherent time complexity of these algorithms.

Algorithm	Parameter-laden	Order-dependent	Non-scalable
ROCK	Yes	No	Yes
CACTUS	Yes	No	No
K-modes	Yes	No[1]	No
COOLCAT	Yes	Yes	Yes
LIMBO	Yes	Yes	No
SUBCAD	Yes	Yes	Yes
AT-DC	No	Yes	No

Table 2.1: Summary of the drawbacks of existing mainstream clustering algorithms for categorical data

[1]But also unstable because of the random initialization.

2.3 Notation and definitions

We will now formally define the notation that will be used throughout this chapter and Chapter 3. Let $T = \{X_1, X_2, \cdots, X_n\}$ be a data set of n objects, where each object is a multidimensional vector of m categorical attributes with domains $D_1, D_2, \cdots, D_m$, respectively. Clustering the data set T consists of dividing the n objects into several groups, i.e., $C = \{C_1, C_2, \cdots, C_K\}$, where each $C_i \neq \emptyset$ $(i = 1, \cdots, K)$ is a cluster, satisfying $C_1 \cup \cdots \cup C_K = T$, $C_i \cap C_j = \emptyset$, for all $i, j = 1, \cdots, K$, $i \neq j$. For each categorical value $v \in D_j$, $p(x_j = v|C_i)$ represents the probability of $x_j = v$ given cluster C_i. In our approach, this probability is estimated by the frequency of v in cluster C_i.

The mathematical definitions of the other symbols used in this chapter are given as follows. I denotes the identity matrix. $\mathbf{1}$ denotes the column vector of all ones in matrix operation. The transpose of matrix A is A^{T}. The trace of matrix A, which is the sum of the elements on the main diagonal, is denoted by $trace(A)$. The vector of row mass of matrix A is denoted as r, where each component r_i is the sum of elements in row i. The vector of column mass, denoted as c, is defined similarly. We also utilize row and column mass matrices in this chapter, which are defined as diagonal matrices with mass elements of r and mass elements of c, respectively. If A is a binary matrix, the row mass and column mass matrices can be written as $diag(AA^{\mathrm{T}})$ and $diag(A^{\mathrm{T}}A)$, respectively.

Table 2.2 summarizes the main notation used in this chapter and Chapter 3.

2.4 MCA calculation on indicator matrix

In this section, we describe how to transform categorical data into an indicator matrix and introduce the MCA calculation on the indicator matrix. The description of MCA calculation is given here on the indicator matrix Z of the whole data set T. In our divisive hierarchical clustering algorithm, the MCA calculation is performed on the indicator matrix $Z^{(p)}$ of each cluster C_p.

The categorical attributes of an input data set are transformed as described below

Symbol	Description
n	Number of objects for clustering
m	Number of attributes
K	Number of clusters
Z	Indicator matrix
$Z^{(p)}$	Indicator matrix of cluster C_p
D_t	Domain of t^{th} attribute
v	Categorical value
J	Total number of categorical values
$\|C_p\|$	Number of objects in cluster C_p
C_p^L,C_p^R	Left and right children (subclusters) of cluster C_p

Table 2.2: Summary of notation used in Chapter 2 and 3

in order to create the indicator matrix, denoted by Z, which is a Boolean disjunctive table. Given the original categorical data set T, we denote the number of values for the t^{th} categorical attribute by $|D_t|$. For each attribute A_t of the original categorical data, there are $|D_t|$ corresponding columns. Therefore, there will be $J = \sum_{t=1}^{m} |D_t|$ columns in Z to represent all the original attributes. Here identical values from different attributes are treated as distinct. The indicator matrix Z is of order $n \times J$, with each element defined as follows:

$$z_{ij} = \begin{cases} 1, & \text{if object } X_i \text{ takes the } j^{\text{th}} \text{ value ;} \\ 0, & \text{otherwise.} \end{cases}$$

Here the j^{th} $(1 \leq j \leq J)$ categorical value corresponds to the j^{th} column of Z. In the remainder of this thesis, we use Z_i to denote categorical object X_i according to the indicator matrix data representation.

For each attribute, we expand its single original column to $|D_t|$ columns, each categorical value taking one column. Of the $|D_t|$ columns, only one column corresponding to the categorical value takes the value 1, while the other columns take the value 0. So the sum of each row of matrix Z is m. This unified data presentation for categorical data also simplifies the following dissimilarity calculation. Furthermore, the role played by each categorical value in forming clusters can be distinguished by

giving them different weights, which can be easily implemented under our new data presentation.

In the example below, a simple data set is used as a scenario to illustrate how the original categorical data is transformed into an indicator matrix. In Table 2.3, there are six categorical objects with three attributes, whose domains are $D_1 = \{a, b, c\}$, $D_2 = \{a, b, c\}$, $D_3 = \{a, b, c\}$, respectively. Table 2.4 illustrates the 9-column indicator matrix of the data set in Table 2.3.

	D1	**D2**	**D3**
1	a	a	a
2	a	c	b
3	a	a	b
4	b	b	a
5	b	b	c
6	c	b	c

Table 2.3: Categorical data set scenario

	D1			**D2**			**D3**		
	a	**b**	**c**	**a**	**b**	**c**	**a**	**b**	**c**
1	1	0	0	1	0	0	1	0	0
2	1	0	0	0	0	1	0	1	0
3	1	0	0	1	0	0	0	1	0
4	0	1	0	0	1	0	1	0	0
5	0	1	0	0	1	0	0	0	1
6	0	0	1	0	1	0	0	0	1

Table 2.4: Indicator matrix of Table 2.3

An indicator matrix is thus a kind of redundant data representation. For each A_t, we expand its single-column representation corresponding to the original attribute to a $|D_t|$-column representation where each column corresponds to one value of A_t. For each object, only one of the $|D_t|$ columns corresponding to the categorical value takes the value 1, while the other columns take the value 0. So the sum of each row of matrix Z is the number of original attributes, m. This unified data presentation for categorical data simplifies the subsequent dissimilarity calculation.

As a special case of categorical data, transactional data can also be transformed to an indicator matrix. Each item of a transaction is analogous to one categorical attribute which can take two values indicating inclusion or non-inclusion of the item. To build the indicator matrix, each such attribute is then transformed to two columns, one corresponding to inclusion and the other to non-inclusion of the item. This is called a symmetric transformation. Thus, if we suppose there are a total of d items in the transaction data base, the indicator matrix of the transaction data will have $2d$ columns rather than d columns (asymmetric transformation of transaction data results in a binary table with d columns). As the result of this transformation, each row of the indicator matrix is guaranteed to have the same sum value, which is d. For example, a transaction $\{a, c, d\}$ over the full item set $\{a, b, c, d, e\}$ is transformed to a row of its indicator matrix with ten columns: the row is [1, 0, 0, 1, 1, 0, 1, 0, 0, 1].

We exploit the standard approach [39] to MCA, i.e., applying a simple CA to the indicator matrix Z. Since Z has a total sum of nm, which is the total number of occurrences of all the categorical values, the correspondence matrix is $P = Z/nm$. Thus the vector of row mass of the correspondence matrix is $r = (1/n)\mathbf{1}$, the row mass matrix is $D_r = (1/n)I$; the column mass matrix is $D_c = (1/nm)diag(Z^{\mathrm{T}}Z)$, the vector of column mass of the correspondence matrix can be written as $c = D_c \times \mathbf{1}$. Under the null hypothesis of independence [39], the expected value of p_{ij} is $r_i c_j$, and the difference between the observation and the expectation, called the residual value, is $p_{ij} - r_i c_j$. Normalization of the residual value involves dividing the difference by the square root of $r_i c_j$. So the standardized residuals matrix is written in matrix notation as:

$$S = D_r^{-1/2}\left(P - rc^{\mathrm{T}}\right)D_c^{-1/2} = \sqrt{n}\left(\frac{Z}{nm} - \frac{1}{n}\mathbf{1}\mathbf{1}^{\mathrm{T}}\mathbf{D_c}\right)D_c^{-1/2}. \tag{2.1}$$

Hence, the singular value decomposition (SVD) to compute the residuals matrix (2.1) is as follows:

$$\sqrt{n}\left(\frac{Z}{nm} - \frac{1}{n}\mathbf{1}\mathbf{1}^{\mathrm{T}}\mathbf{D_c}\right)D_c^{-1/2} = U\Sigma V^{\mathrm{T}}, \tag{2.2}$$

where $U^{\mathrm{T}}U = V^{\mathrm{T}}V = I$. Σ is a diagonal matrix with singular values in descending order: $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_s > 0$, where s is the rank of the residuals matrix. The

columns of U are called the left singular vectors, and those of V, the right singular vectors. The left singular vectors U give us the scale values for the n objects, while the right singular vectors V give us the scale values for the J categorical values. In CA, they are called principal coordinates of rows and principal coordinates of columns, denoted as follows:

Principal coordinates of rows:

$$F = D_r^{-1/2} U \Sigma$$

Principal coordinates of columns:

$$G = D_c^{-1/2} V \Sigma$$

The principal coordinates of the rows and columns can be plotted in the same coordinate system, as shown in Section 2.6.1. This graphical representation, called a symmetric map, illustrates the pattern of association between rows and columns. To avoid numerical overflow caused by $1/nm$ when n and m are very large values, the standardized residuals matrix (2.1) can be written in an equivalent form as:

$$S = \left(Z - \frac{1}{n} \mathbf{1}\mathbf{1}^{\mathrm{T}} \hat{\mathbf{D}}_{\mathbf{c}} \right) \left(m \hat{D}_c \right)^{-1/2}, \tag{2.3}$$

where $\hat{D}_c = diag\left(Z^{\mathrm{T}} Z\right)$. The equivalent transformation also simplifies the calculation.

The MCA calculation on the residuals matrix (2.3) provides an effective way to measure the association among the objects. The sum of squared elements of the standardized residuals matrix, called total inertia in correspondence analysis, is as follows:

$$\sum_i \sum_j s_{ij}^2 = \sum_i \sum_j \frac{(z_{ij} - z_{\cdot j}/n)^2}{m z_{\cdot j}} = \frac{1}{mn} \sum_i \sum_j \frac{(z_{ij} - z_{\cdot j}/n)^2}{z_{\cdot j}/n}, \tag{2.4}$$

where $z_{\cdot j} = \sum_i z_{ij}$, $\sum_j \frac{(z_{ij} - z_{\cdot j}/n)^2}{z_{\cdot j}/n}$ is the Chi-square statistic measuring the association between each object Z_i and the set of objects T from the perspective of correspondence

analysis, while from the perspective of clustering, it is a distance measure between Z_i and T, i.e., the Chi-square distance $d_{Chi}(Z_i, T)$. Thus Formula (2.4) can be written as follows:

$$\sum_i \sum_j s_{ij}^2 = \frac{1}{mn} \sum_i d_{Chi}(Z_i, T). \tag{2.5}$$

Thus the total inertia can be explained as the average of the Chi-square distances between objects and the data set. The total inertia can also be expressed in the following form:

$$\sum_i \sum_j s_{ij}^2 = trace(SS^{\mathrm{T}}) = trace(\Sigma^2) = \sum_i^s \sigma_i^2 \tag{2.6}$$

From (2.5) and (2.6) we can see that the average of the Chi-square distances is equivalent to the sum of the eigenvalues from the MCA calculation. In MCA, an eigenvalue σ^2 represents the amount of inertia that reflects the relative importance of the transformed dimension. The first dimension always explains the largest portion of the variance, and the second explains the largest portion of the remaining unexplained variance, etc. In divisive hierarchical clustering, the goal of splitting a cluster is to lower the variance within the resulting subclusters, which consists of lowering the average of the Chi-square distance, or equivalently, the sum of the eigenvalues from the MCA standpoint. We will further explain the relationship between MCA and clustering categorical data in Section 2.6.1.

2.5 Optimization perspective for clustering categorical data

In this section, we formalize the general problem of clustering n categorical objects into K groups, which is defined in Section 2.3, from an optimization perspective. We propose an objective function, and calculate the cluster center by optimizing the objective function.

We exploit Chi-square distance between a single object and a set of objects to

define the objective function. We argued previously that in some situations, the similarity defined between a single object and a set of objects is more meaningful than pairwise similarity, especially when there is no significant comparison that can be used to define pairwise similarity in cluster analysis [7, 88], as in the example given in Section 2.1. Furthermore, MCA calculation on the indicator matrix involves a measure of the Chi-square dissimilarity between a single object and a set of objects, so we set the objective of clustering categorical data set T into K groups so as to minimize the following objective function, i.e., the sum of Chi-square error (SCE):

$$SCE = \sum_{k=1}^{K} \sum_{Z_i \in C_k} d_{Chi}(Z_i, C_k), \tag{2.7}$$

where $d_{Chi}(Z_i, C_k)$ is the Chi-square distance between object Z_i and cluster C_k, which is defined as follows:

$$d_{Chi}(Z_i, C_k) = \sum_{j} \frac{(z_{ij} - \mu_{kj})^2}{\mu_{kj}}. \tag{2.8}$$

Here $\mu_{kj} (1 \leq j \leq J)$ is the j^{th} element of the cluster center of cluster C_k, and object Z_i is in cluster C_k.

The cluster center can be determined by optimizing the objective function (2.7), from which, the cluster center of cluster C_k in (2.8) is defined as the square root of the frequency of the categorical value in the cluster, i.e.,

$$\mu_{kj} = \sqrt{p(v_j|C_k)} = \left(\frac{z_j}{|C_k|}\right)^{1/2}, \tag{2.9}$$

where v_j is the j^{th} categorical value, $z_j = \sum_{Z_i \in C_k} z_{ij}$. It is worth noting that when $\mu_{kj} = 0$, $(z_{ij} - \mu_{kj})^2$ must be zero as z_{ij} must be zero: in this case, $(z_{ij} - \mu_{kj})^2 / \mu_{kj} = 0$. We give the derivation below. It shows how the cluster center defined in (2.9) can be mathematically derived when the distance measure is defined as in (2.8) and the objective is to minimize the SCE defined in (2.7).

Specifically, the SCE function is written as

$$SCE = \sum_{k=1}^{K} \sum_{Z_i \in C_k} \sum_{j=1}^{J} \frac{(z_{ij} - \mu_{kj})^2}{\mu_{kj}} \tag{2.10}$$

We solve for the t^{th} element of p^{th} cluster center $\mu_{pt} (1 \leq t \leq J)$ which minimizes equation (2.10) by differentiating the SCE, setting it equal to 0. The derivation is as follows:

$$\begin{aligned}
\frac{\partial}{\partial \mu_{pt}} SCE &= \frac{\partial}{\partial \mu_{pt}} \sum_{k=1}^{K} \sum_{Z_i \in C_k} \sum_{j=1}^{J} \frac{(z_{ij} - \mu_{kj})^2}{\mu_{kj}} \\
&= \sum_{k=1}^{K} \sum_{Z_i \in C_k} \sum_{j=1}^{J} \frac{\partial}{\partial \mu_{pt}} \frac{(z_{ij} - \mu_{kj})^2}{\mu_{kj}} \\
&= \sum_{Z_i \in C_p} \sum_{j=1}^{J} \frac{\partial}{\partial \mu_{pt}} \frac{(z_{ij} - \mu_{pj})^2}{\mu_{pj}}
\end{aligned}$$

Under the hypothesis of independence of the dimensions (the same as the null hypothesis of independence in MCA), we get

$$\begin{aligned}
\frac{\partial}{\partial \mu_{pt}} SCE &= \sum_{Z_i \in C_p} \frac{\partial}{\partial \mu_{pt}} \frac{(z_{it} - \mu_{pt})^2}{\mu_{pt}} \\
&= \sum_{Z_i \in C_p} \frac{2(\mu_{pt} - z_{it})\mu_{pt} - (\mu_{pt} - z_{it})^2}{\mu_{pt}^2} \\
&= \sum_{Z_i \in C_p} \left(1 - \frac{z_{it}^2}{\mu_{pt}^2}\right)
\end{aligned}$$

$$\sum_{Z_i \in C_p} \left(1 - \frac{z_{it}^2}{\mu_{pt}^2}\right) = 0 \Rightarrow \mu_{pt}^2 = \frac{1}{|C_p|} \sum_{Z_i \in C_p} z_{it}^2 \Rightarrow \mu_{pt} = \left(\frac{1}{|C_p|} \sum_{Z_i \in C_p} z_{it}^2\right)^{1/2}$$

As z_{it} is a Boolean variable having value 0 or 1, $z_{it}^2 = z_{it}$, $\mu_{pt} = \left(\frac{1}{|C_p|} \sum_{Z_i \in C_p} z_{it}\right)^{1/2}$. Thus, the prototype of the cluster in categorical data is represented by the square root of the mean of the indicator matrix of the cluster.

2.6 The DHCC algorithm

A detailed description of DHCC is given in this section. In contrast to the agglomerative approach, DHCC starts with an all-inclusive cluster containing all the categorical objects, and repeatedly chooses one cluster to split into two subclusters. A binary tree is employed to represent the hierarchical structure of the clustering results, in a way similar to that used in conventional hierarchical clustering algorithms [26, 95]. Additionally, in this section, we explain why DHCC can discover clusters embedded in subspaces.

In DHCC, splitting a cluster C_p involves finding a suboptimal (if not optimal) solution to the optimization problem on the data set C_p with K=2. The overall scheme of DHCC is given in the algorithm in Figure 2.1. The core of the DHCC algorithm is the bisection procedure, which consists of two phases, preliminary splitting (step 3) and refinement (step 4). The algorithm iteratively chooses a leaf cluster to split, unless no leaf cluster can be split to further improve clustering quality. The quality measure will be presented in Section 2.6.3.

Algorithm: Scheme of DHCC

1. Transform the original categorical data into indicator matrix Z.
2. Initialize a binary tree with a single root holding all the objects.
3. Choose one leaf cluster C_p to split into two clusters C_p^L and C_p^R based on MCA calculation on the indicator matrix $Z^{(p)}$.
4. Iteratively refine the objects in clusters C_p^L and C_p^R.
5. Repeat steps (3) and (4) until no leaf cluster can be split to improve the clustering quality.

Figure 2.1: Scheme of DHCC

2.6.1 Preliminary splitting

In each bisection step, we initialize the splitting based on MCA. From formulas (2.5) and (2.6) we can see that minimizing the objective function (2.7) in a bisection involves maximally decreasing the total inertia of the standardized residuals matrices of the two resulting subclusters. As the first dimension of the transformed space based on MCA accounts for the largest proportion of the total inertia, splitting a cluster based on the first dimension can efficiently generate a preliminary bisection toward the optimization of the objective function.

Preliminary splitting is performed as follows. To bisect cluster C_p with $|C_p|$ objects, we apply MCA on the indicator matrix $Z^{(p)}$ of order $|C_p| \times J$ from the $|C_p|$ objects to get the principal coordinates of rows, i.e., $F^{(p)}$. The object Z_i whose first coordinate $F_{i1}^{(p)} \leq 0$ (or $U_{i1}^{(p)} \leq 0$) goes to the left child of C_p, which is denoted by C_p^L, and the object Z_i whose first coordinate $F_{i1}^{(p)} > 0$ (or $U_{i1}^{(p)} > 0$) goes to the right child of C_p, which is denoted by C_p^R. In this phase, MCA plays an important role in analyzing the data globally, and the variance and data distribution of the objects are thus taken into account in the preliminary bisection.

Why do we use only one dimension of the transformed space based on MCA to perform the preliminary bisection? Apart from the computational efficiency consideration, our concern is that other (less significant) dimensions may account for variance that is unlikely to be of interest in clustering, especially for the dimensions of lower inertia. Inappropriate involvement of these dimensions may result in adverse preliminary splitting. Take the data for the scenario in Table 2.3 for example. Clearly, the first 3 objects should be grouped together, as they are associated by the first attribute, having the common value 'a'; and the last 3 objects should likewise be grouped together, as they are associated by the second attribute, having the common value 'b'. The variance for the third attribute should be discarded in the clustering analysis. The symmetric map of the scenario data is given in Figure 2.2. The x-axis accounts for 46.2% of the total inertia, and the two clusters can be separated correctly on this dimension. The categorical value 'a' on the third attribute has a large value on the y-axis, which accounts for 25% of the total inertia; however, it does not contribute to distinguish the two clusters as it appears once in both clusters. Division

according to the most significant dimension is very simple. It will be shown in our experiment that the simple preliminary division works well for clustering categorical data in DHCC.

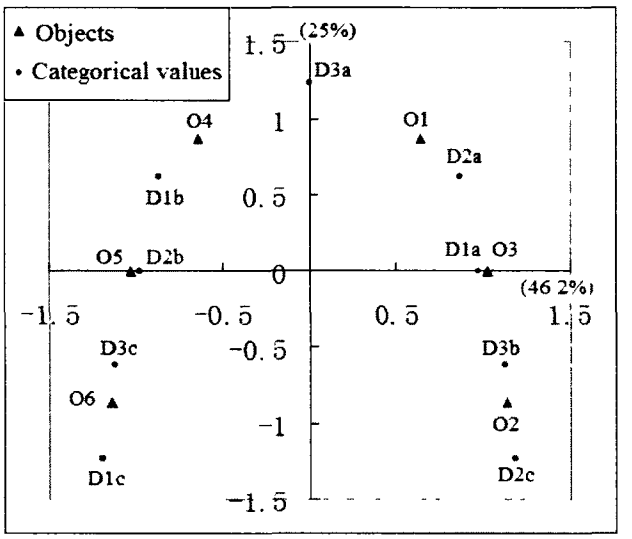

Figure 2.2: Symmetric correspondence analysis map of scenario data

2.6.2 Refinement

The refinement phase attempts to improve the quality of the bisection by relocating the objects from the cluster being split. After the preliminary bisection, for each object from the parent cluster C_p, the refinement phase tries to improve the splitting quality by finding which subcluster, C_p^L or C_p^R, is more suitable.

For computational efficiency, some cluster features associated with each cluster C_p are maintained. One such feature is the J-dimensional vector of occurrence numbers of all the categorical values, denoted as o_p; the other is the number of objects in the cluster, i.e., $|C_p|$. Each element of o_p, conveniently represented here by z_j, is the occurrence number of the j^{th} categorical value. The j^{th} element of cluster center can be calculated as $\mu_{pj} = \sqrt{z_j/|C_p|}$, which in turn can be used to calculate the Chi-square distance if the object Z_i is in cluster C_p. If Z_i is not in C_p, then the center for Chi-square distance is computed as $\mu_{pj} = \sqrt{(z_j + z_{ij})/(|C_p| + 1)}$, since, similar

to the Chi-square statistic in (2.4), the Chi-square distance in (2.8) also requires that the individual object be a member of the set. Thus when calculating the dissimilarity between an object Z_i and cluster C_p, where Z_i is not in C_p, we should temporarily place Z_i in cluster C_p, using $o_p + Z_i$ as category occurrence feature and $|C_p| + 1$ as the number of objects in the cluster.

The iterative refinement of C_p^L and C_p^R proceeds as in the algorithm in Figure 2.3.

Algorithm: Refine the preliminary bisection

1. Calculate the cluster features of C_p^L and C_p^R, i.e., o_p^L, $|C_p^L|$ and o_p^R, $|C_p^R|$.
2. For each object Z_i in C_p^L
   $o_p^R = o_p^R + Z_i$; $|C_p^R| = |C_p^R| + 1$;
   if $d_{Chi}(Z_i, C_p^R) < d_{Chi}(Z_i, C_p^L)$, move Z_i to C_p^R.
   $o_p^R = o_p^R - Z_i$; $|C_p^R| = |C_p^R| - 1$;
   For each object Z_i in C_p^R
   $o_p^L = o_p^L + Z_i$; $|C_p^L| = |C_p^L| + 1$;
   if $d_{Chi}(Z_i, C_p^L) < d_{Chi}(Z_i, C_p^R)$, move Z_i to C_p^L.
   $o_p^L = o_p^L - Z_i$; $|C_p^L| = |C_p^L| - 1$;
3. Update the cluster features of o_p^L, $|C_p^L|$ and o_p^R, $|C_p^R|$.
4. Repeat steps (2) and (3) until the membership no longer changes.

Figure 2.3: Algorithm of refining the preliminary bisection

2.6.3 Termination of splitting process

When to terminate the splitting process is one of the key issues in divisive hierarchical clustering algorithms. Our aim is to design a parameter-free algorithm, so the conventional methods do not work for us, as either the number of clusters K or an ad-hoc stopping threshold is needed in these methods [26, 95]. Recent studies have proposed a "knee" approach to find the right number of clusters, which involves examining the 'number of clusters vs. clustering evaluation metric' graph to spot the

number at which there is a knee [71]. Such an approach requires considerable extra splitting to locate where the knee is. Additionally, it depends on the data distribution (in numeric data) and the right choice of evaluation metric [77].

We seek a global clustering quality measure for determining whether a cluster should be split or not. The SCE objective function in Section 2.5 does not work for this purpose due to a property it shares with the SSE (sum of squared error) in the K-means, i.e., the optimal value of the objective function decreases monotonically with increasing K [26]. In fact, both the SCE and SSE objective functions measure only the "compactness" of the clusters. Therefore, using the SCE objective function as a validity index would result in creation of the whole hierarchical tree, as DHCC always improves the compactness when it splits a cluster. A suitable validity index should make a good trade-off between compactness and separation [76].

In this chapter, we propose a practical validity index based on a combination of compactness, measured by entropy, and separation, measured by cluster size. The compactness (also called homogeneity) of a cluster is closely linked to the concept of entropy [7]; thus we use the information gain, i.e., the decrease in the uncertainty about the occurrences of the categorical values when separating a subset from the whole data, to measure the compactness of cluster C_p. The compactness of C_p is measured as follows:

$$Q(C_p) = \frac{|C_p|}{n} \left(\sum_{j=1}^{J} p(v_j|C_p) \log p(v_j|C_p) - \sum_{j=1}^{J} p(v_j|T) \log p(v_j|T) \right) \tag{2.11}$$

where $p(v|T)$ is the frequency of v over the whole data set. Splitting a cluster always increases the homogeneity of the two resulting subclusters, as $Q(C_p) \leq Q(C_p^L) + Q(C_p^R)$ always holds, [2] but in turn decreases the separation between the two subclusters when the splitting goes deeper. Thus we incorporate cluster size as a separation factor to balance compactness and separation in the validity index, which is defined

[2]This is intuitive, as merging clusters always increases disorderliness. A strict proof is provided in [18].

as:

$$Q(C) = \sum_{C_i \in C} \frac{|C_i|}{n} Q(C_i) \tag{2.12}$$

The validity index in (2.12) is used to implement step 5 of the algorithm in Figure 2.1 (scheme of DHCC). All the leaf clusters are subject to splitting. The leaf clusters can be chosen either through a depth-first or a breadth-first search scheme. If splitting a leaf node leads to improvement of clustering quality, the split is justified; otherwise, the leaf node is a final cluster. The splitting is terminated when no leaf cluster can be split to increase the global clustering quality. The clusters which are not involved in the splitting, whose contribution remains the same before and after it, can be ignored in determining whether the splitting is justified. That is, if the splitting of cluster C_p satisfies $Q(\{C_p\}) < Q(\{C_p^L, C_p^R\})$, it is justified; otherwise, it is not.

2.6.4 Subspace clustering

In this section, we explain why DHCC is capable of discovering clusters embedded in different subspaces. Variable selection and weighting have been widely used for subspace clustering for numeric data [3, 13, 44]. In what is known as soft projected clustering [59], each variable is assigned a weight for each cluster in order to weigh their relevance/contribution to the cluster structure. Determining meaningful and effective values for these weights is a major difficulty that the soft projected clustering algorithms have to overcome.

In DHCC, categorical attributes are also weighted. This is accomplished naturally by the use the Chi-square distance. The weighting scheme in the Chi-square distance differs from the weighting methods for numeric data in that the weights in the Chi-square distance are an intrinsic part of the distance measure. Moreover, the weights in DHCC act directly on categorical values rather than on the attributes. This allows us to more precisely control the effect of each attribute on the cluster structure formatting. Indeed, if we view the Chi-square distance as a weighted Euclidean distance, then the weight on the j^{th} value (dimension of Z) in cluster C_i is μ_{ij}^{-1}. Remember that μ_{ij} is the cluster center of cluster C_i, which is the square root of the frequency of the j^{th} value. Thus, the dimension corresponding to the rarer value

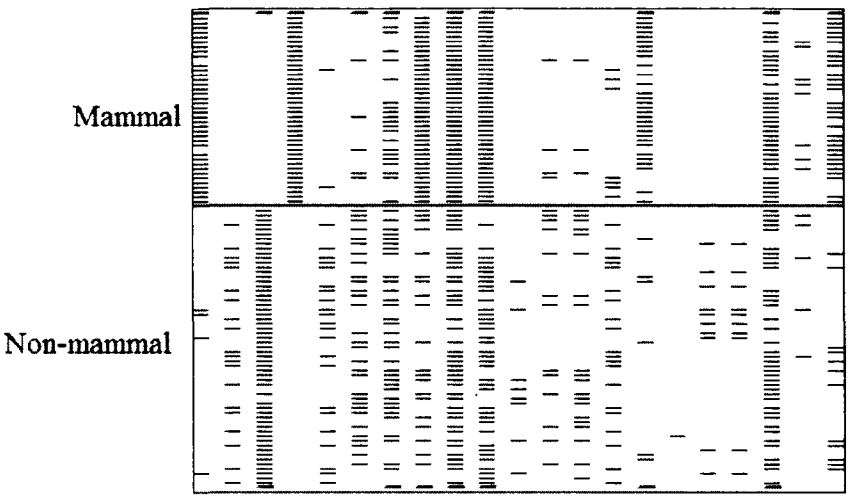

Figure 2.4: Incidence matrix of bisection of *Zoo*

is weighted more heavily. In other words, DHCC favors frequent categorical values while tending to reject rare values.

To illustrate how DHCC is capable of subspace clustering and how the refinement step improves the preliminary bisection, we will now look at a simple example, using the *Zoo* data from the UCI Machine Learning Repository [3]. The *Zoo* data set has 101 objects with 16 categorical attributes, 15 of them Boolean-valued and 1 multi-valued. The data set has class labels dividing the animals into 7 different families. The 15 Boolean-valued attributes indicate whether an object does or does not possess such features as having hair, having feathers, laying eggs, being aquatic, etc. The multi-valued attribute is the number of legs, and the domain is {0, 2, 4, 5, 6, 8}. So, the number of categorical values is J=36. The clusters resulting from one division are illustrated by the incidence matrix graph shown in Figure 2.4. The matrix has 21 columns, representing the 15 Boolean attributes and the 6 values for number of legs. If the animal represented by row i has the feature represented in column j, then there is a dash at the position (i,j).

Figure 2.4 illustrates the final results of DHCC in dividing the whole data set into two clusters. In the preliminary step of this division, the 41 mammalian animals

[3]UCI Machine Learning Repository. Available: `http://archive.ics.uci.edu/ml/`

and the tuatara (which belongs to the reptile family) are grouped together, and the other 59 animals are grouped together. The refinement step reassigns the tuatara from the mammal cluster to the non-mammal cluster. The top part of the incidence matrix, down to the horizontal solid line in the middle, represents the mammal family (cluster), whereas the bottom part represents the non-mammal family (cluster). And the tuatara, now in non-mammal, is represented in the last row. From Figure 2.4 we can see that the third (if lays eggs) and fourth (if produces milk) attributes are strongly associated with the subspaces of the structure of both mammal and non-mammal clusters. The mammal cluster "rejects" the tuatara, in the refinement step, because it takes rare values on the third and fourth attributes, i.e., lays eggs but does not produce milk. After refinement, the platypus, represented in the first row, remains in the mammal cluster although it lays eggs because it has hair (first attribute) and produces milk, which are strongly rejected by the non-mammal cluster.

2.6.5 Algorithm analysis

The time complexity of the DHCC algorithm is linear with respect to the number of objects. For each bisection step, preliminary division consumes more time than refinement. Suppose that the final clustering tree is balanced, the time complexity of refinement operation for the whole algorithm is $O(\psi nJ \log K)$, where ψ is the average number of iterations for refinement at all bisections. Thus the refinement operation has linear time complexity with respect to n when other factors ψ and J are treated as constants. The time consumed for preliminary division is taken primarily by the SVD computation. Determining the exact SVD of an $n \times J$ matrix has a time complexity of $O(nJ \min(n, J))$. Faster SVD algorithms have been proposed in recent years [14, 29], lowering the complexity to $O(nJr)$ for $r \leq \sqrt{\min(n, J)}$, where r is the rank of the indicator matrix. So each preliminary division is scalable to very large data set, where r mainly depends on J. Finally, combining the linear time complexities for both phases of bisection, strictly speaking, with some restriction on ψ and J, the time complexity of the whole DHCC algorithm is linear with respect to the number of the objects for clustering, so DHCC is scalable for very large data sets.

The DHCC algorithm is independent of the order in which the data is processed. Neither the preliminary splitting nor the refinement phase involves any operation that depends on the order. This is not the case for many other algorithms. The incremental (re)location [4] process used in [7, 15, 34], in which the centroid or model of the cluster is updated immediately after relocation of an object, results in the clustering results being sensitive to the processing order. Different processing orders cause the clustering to evolve differently, which may ultimately lead to different clustering results. Besides the (re)location process, other operations also cause the algorithm to be sensitive to the processing order, such as the splitting process in AT-DC [15] and the initialization process in COOLCAT [7] and SUBCAD [34]. In DHCC, in the preliminary splitting phase, the coordinates of rows, which are the output of the MCA calculation used to perform preliminary splitting, are not affected by the order. The refinement phase runs like the traditional K-means algorithm; that is, the cluster centers are updated at the end of each iteration, so the iterative refinement procedure is also independent of the order. The DHCC algorithm is thus order-independent.

2.7 Experimental results

In this section, we perform an extensive comparison of DHCC with some mainstream algorithms, on both real and synthetic data. With the source code of the SVD algorithm available on the Web [5]. DHCC is quite simple to implement. As there is no parameter imposed in the algorithm and it is order-independent, DHCC can be evaluated objectively. We compare DHCC with five mainstream algorithms: K-modes [72], COOLCAT [7], SUBCAD [34], CACTUS [6] [35] and AT-DC [15]. It is reported in [15] that AT-DC outperforms most of the existing algorithms, so the comparison with AT-DC extends our comparison to other existing algorithms. As both the real-life and synthetic data we used have class labels, the algorithms can be

[4]The COOLCAT [7] algorithm terminates when all the objects are clustered without calling the relocation process.

[5]ALGLIB, SVD code. Available: `http://www.alglib.net/matrixops/general/svd.php`

[6]As CACTUS was unable to generate clusters on almost all the data sets in our experiment, its results do not appear in the comparison.

evaluated objectively on how well they discover the natural structure of the data.

2.7.1 Quality measures

Evaluating the quality of clustering results is a challenge, as there is no consistent and unique way (such as using the precision measure in classification) to do it [77, 84]. As an exploratory data analysis method, each different clustering algorithm defines its own type of cluster, which leads to various evaluation measures. Often, these measures conflict with each other: a good clustering according to one measure may be evaluated poorly by the other measures. The existing quality measures are classified into the following three types [77]:

Unsupervised: when the clustering results are evaluated without referring to external information. The measure tends to determine how closely related the objects in a cluster are, or how well-separated a cluster is from other clusters;

Supervised: when the clustering results are evaluated according to an external structure. The measure determines how well the clustering structure matches externally supplied class labels;

Relative: when the clustering results from different algorithms on the same data set are compared.

In our performance evaluation, we adopt measures of all three types. The supervised measure adopted is normalized mutual information (NMI). Wu et al. (2009) have established empirically and theoretically that *NMI*, which is equivalent to normalized variation of information, is one of the most suitable quality measures. *NMI* is more appropriate on data with imbalanced distributions, does not necessarily improve when the number of clusters increases (as purity and entropy do), and additionally, is sensitive to change in the clustering results (making it suitable to measure the stability of the clustering algorithm). The unsupervised measure named Category Utility (CU), which is commonly used to evaluate the clustering quality of categorical data [5, 7, 27], is also used in our performance evaluation. Definitions of these measures are given below.

NMI estimates how much information a cluster label can tell about the class, i.e., the extent to which the clustering structure exactly matches the external classification. It is defined as

$$NMI = \frac{I(O;P)}{(H(O)+H(P))/2} \tag{2.13}$$

where O is the random variable denoting the external class label of the objects and P is the random variable denoting the cluster label generated by the clustering algorithm for the same objects; $I(O;P) = H(O) - H(O|P)$ is the mutual information between O and P; $H(O)$ and $H(P)$ are the Shannon entropy of O and P respectively, and $H(O|P)$ is the conditional entropy of O given P. Normalization of mutual information by the average entropy of the variables of class and cluster labels keeps the value of *NMI* between 0 and 1. The ideal value for *NMI* is 1, meaning an exact match of the clustering structure and the external class structure, i.e., the K is the same as the number of classes, and all the clusters are pure with respect to the distribution of the classes. If the NMI value is 0, then the distribution of the objects in each cluster will tend to be the same as that in the whole data set, which means that clustering has been done randomly.

Category Utility (CU) is defined as

$$CU(C) = \sum_{C_i \in C} \frac{|C_i|}{n} \sum_{j=1}^{m} \sum_{v \in D_j} \left[(p(x_j = v|C_i))^2 - (p(x_j = v|T))^2 \right] \tag{2.14}$$

where $p(x_j = v|C_i) > 0$. The CU measures the difference between the frequency of the categorical value in a cluster and the frequency of the same values in the whole set of objects. A higher CU value indicates a better clustering result, as a good partition of the data should increase the intra-cluster homogeneity, that is, the sum of the frequency of the categorical values in clusters should be higher than the sum of the frequency of the same values over in the whole data set. The optimal value of CU increases monotonically with increasing K ($K = |C|$); hence, different clustering results should have the same K for comparison via CU to make sense.

2.7.2 Synthetic data

We used synthetic data sets to evaluate both the clustering quality and the scalability of DHCC. The synthetic data sets were obtained using a synthetic data generator available on the Web [7]. We generated four different data sets with different numbers of attributes (m=5, 10, 15, 20), with J=9, 16, 38, 57 respectively; each data set contains 1000 objects belonging to three different classes. The class labels are generated by conjunctive rules formed in the dimensionality of the subspace, i.e., $x_{s1} = a_1 \wedge x_{s2} = a_2 \wedge \cdots \Rightarrow Class = s$, $s \in \{1, 2, 3\}$. The class rules of the four data sets are shown in Table 2.5. To discover the true clusters, the clustering algorithm should be capable of subspace clustering.

$m = 5$	$C_1 \Leftarrow x_{1,2,3,4,5} = (b, a, a, c, a)$
	$C_2 \Leftarrow x_{1,2,3,4,5} = (b, a, a, a, b)$
	$C_3 \Leftarrow x_{1,2,3,4,5} = (a, a, a, b, a)$
$m = 10$	$C_1 \Leftarrow x_{1,4,5,7,9,10} = (a, a, a, e, b, a)$
	$C_2 \Leftarrow x_{6,7,9,10} = (a, e, a, a)$
	$C_3 \Leftarrow x_{1,3,4,5,6,7,8,9,10} = (a, a, a, a, b, e, a, a, a)$
$m = 15$	$C_1 \Leftarrow x_{4,6,7} = (b, a, b)$
	$C_2 \Leftarrow x_{2,3,4,5,7,9,10,11,14,15} = (b, b, b, a, a, c, a, b, a, a)$
	$C_3 \Leftarrow x_{1,2,4,5,7,8,12} = (a, b, c, b, b, a, a)$
$m = 20$	$C_1 \Leftarrow x_{9,13,15,17} = (d, b, b, a)$
	$C_2 \Leftarrow x_{5,9,18,19} = (b, e, d, a)$
	$C_3 \Leftarrow x_{1,6,8,9,13,18,19} = (a, b, c, a, e, a)$

Table 2.5: Class rules of synthetic data

The clusters discovered by DHCC on the four synthetic data sets exactly fit the natural classification, so the *NMI* values of the four clustering results are 1. The comparisons with AT-DC, COOLCAT, SUBCAD and K-modes in terms of *NMI* are shown in Figure 2.5; those by *CU* are shown in Table 2.6. For the K-modes, COOLCAT and SUBCAD algorithms, which need K to be supplied, we simply set $K = 3$. For COOLCAT and SUBCAD, to make sure they produce the best results, we used the entire set as a sample. AT-DC uses a strategy to deal with the problem of order-dependence, and the strategy was exploited in our implementation. As these

[7]Data Generator. Available: http://www.datasetgenerator.com/.

algorithms are either initialization-dependent or order-dependent, we ran each of them 10 times with different initializations and processing orders, the 10 processing orders being the original order, the reversed original order, the order associated with class labels and seven random orders. As their results are unstable on different orders and initializations, we give three results on each data set: the best, the worst and the average for each algorithm. In Figure 2.5, the vertical line covers the range from the minimum to the maximum value of NMI, and the horizontal line represents the average value. In Table 2.6, the three values of CU appear in the first, second and third rows in the section for each data set. If the algorithm is stable on a data set, the corresponding results are given in one row. Because the two quality measures, NMI and CU, are not always consistent with each other, the clustering output that yields the best NMI value may not be the same one that yields the best CU value for a given algorithm. The same is true for the worst value. The average result is the mean of the measured values from the 10 runs.

	m=5	m=10	m=15	m=20
DHCC	**2.226**	**1.307**	**3.042**	**2.877**
AT-DC	**2.226**	n/a	3.183(K=4)	3.139(K=5)
COOLCAT	**2.226**	1.307	2.887	2.756
		0.786	1.423	1.056
		0.997	2.625	2.343
SUBCAD	**2.226**	1.307	3.042	2.877
		0.288	1.927	0.970
		0.976	2.931	2.438
K-modes	2.226	1.236	1.866	1.672
	0.773	0.339	0.968	1.468
	1.62	0.833	1.755	1.576

Table 2.6: Comparison of DHCC in terms of CU on synthetic data

From Figure 2.5 and Table 2.6, it can be seen that DHCC outperforms the other algorithms on the synthetic data. AT-DC is unstable on data set m=10: even though it adopts a strategy to deal with order-dependence, it generates quite different numbers of clusters from different processing orders, varying from 2 to 8. It is not meaningful to compare and calculate the average of these CU values for different K, as CU

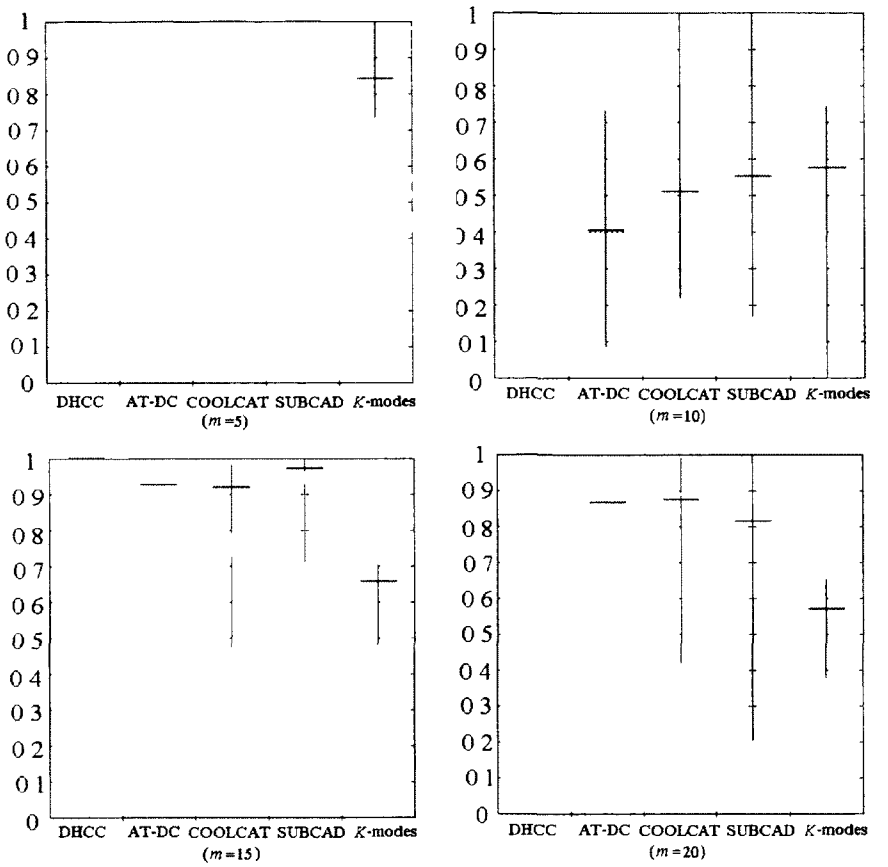

Figure 2.5: Comparison of DHCC in terms of *NMI* on synthetic data

is closely related to K: hence the "n/a" in the cell for AT-DC on data set m=10 in Table 2.6. AT-DC obtains the highest CU on data sets m=15 and m=20; this is because AT-DC generates 4 clusters on m=15 and 5 clusters on m=20, and CU tends to increase when more clusters are generated. All the algorithms except the K-modes can stably discover the true classification on data set m=5, because of its low dimensionality and the simple structure of the data (the 3 clusters are formed on the whole space). AT-DC, COOLCAT, SUBCAD and K-modes perform poorly on data set m=10 in terms of both average quality and stability because the classes are defined on subspaces with quite different dimensionalities, as can be seen in Table 2.5: Class $C2$ is formed in a small subspace, while Class $C3$ is defined over almost the whole space. SUBCAD, which is designed for subspace clustering, is the only

algorithm except DHCC that can correctly discover the true structure of all the synthetic data sets. AT-DC is also capable of subspace clustering [15], generating pure clusters on data sets m=15 and m=20, but it finds more than the true number of clusters (classes). Both SUBCAD and COOLCAT are very sensitive to the processing order; for these two algorithms, an 'improper' order can lead to very poor clustering in our experiment. Furthermore, AT-DC is unstable in some cases (such as the data set m=10) even though it adopts a strategy to deal with order-dependence. In most cases, the K-modes obtains poor clustering results. Additionally, it is very sensitive to the initialization in our experiments. As for CACTUS, no cluster can be generated on the four data sets when the support threshold ratio α ($\alpha > 1$) is set to 2, or a value between 2 and 3, as suggested in Ganti et al. (1999). When α is set to 1.2, three clusters are generated on data set m=5, exactly fitting the external classes; however, no cluster is generated on the other three data sets; this is because the clusters are formed in subspaces in these three data sets.

The following scalability test on DHCC confirms that DHCC has linear time complexity with respect to the number of objects for clustering and the total number of categorical values. Note that the complexity of DHCC involves a product of n and J; here we consider a single dimension while treating the other as a constant. DHCC was implemented in C++ and executed on an Intel Core 2 Duo desktop processor with 2 Gbytes of memory and 2.66 GHz of clock speed. To test the scalability with respect to the number of objects, we generated four data sets with m=10, complying with the rules of the synthetic data set m=10. The sizes of the four data sets are 10^3, 10^4, 10^5 and 10^6, respectively. To test the scalability with respect to J, we generated four other data sets with m=10, 100, 1000 and 10000 and $J = 2m$; each of these data sets contains 1000 objects belonging to two different classes. We also compared DHCC with the scalable algorithm AT-DC for execution time. The AT-DC used here is the version without object sorting to deal with order sensitivity.

Figure 2.6 presents the efficiency of DHCC and AT-DC on data sets of varying size. It can be seen that the gap between DHCC and AT-DC widens as the size of the data set increases. DHCC is more efficient than AT-DC when the data set is really large (for example, when it reaches 10^6 as in our experiment). This is mainly due to

the fact that the number of iterations of AT-DC in the cluster-splitting step increases linearly with increasing data size, as shown in Cesario et al. (2007), whereas in DHCC, the number of iterations in the refinement step is independent of the data size, as the majority of objects are assigned to the correct subcluster in the preliminary splitting by MCA. Another fact underlying the linear performance in Figure 2.6 is that DHCC still generates three clusters, exactly fitting the true class structure, on the four large sets. Thus the influence of K on execution time, which is $\log K$, is a constant for each case. Figure 2.6 demonstrates that DHCC is perfectly scalable with data size (treating m as a constant).

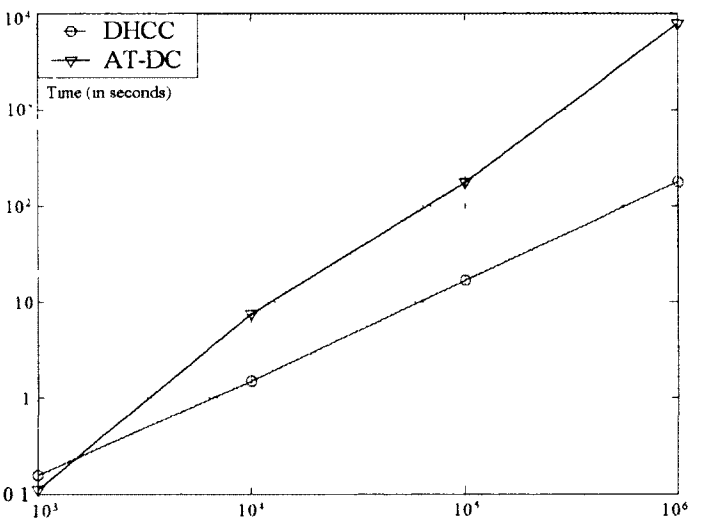

Figure 2.6: Scalability with respect to data size

Figure 2.7 demonstrates the scalability of DHCC with respect to dimensionality. As DHCC operates on the indicator matrix in both the preliminary splitting and the refinement steps, the time complexity with respect to m is directly linked to J. Fortunately, the domain size of categorical attribute is normally not very large; otherwise, it typically does not contain useful information for classification [35]. Figure 2.7 shows that both DHCC and AT-DC behave linearly with respect to m, and AT-DC performs better than DHCC in terms of scalability with dimensionality. The linear time complexity with respect to m (treating n as a constant) makes it possible to apply DHCC to extremely high-dimensional categorical data (such as the *Internet*

Ads data in Section 2.7.3).

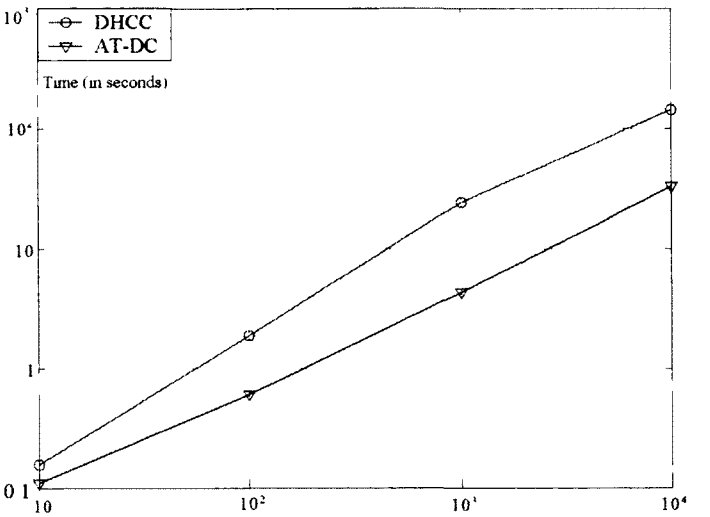

Figure 2.7: Scalability with respect to dimensionality

2.7.3 Real-life data

DHCC was further evaluated on real-life data. Besides the *Zoo* data set, the other three data sets used for evaluation are the Congressional Voting Records, Mushroom and Internet Advertisements sets from the UCI Machine Learning Repository. A description of each data set follows.

The Congressional Voting Records data set (*Votes*) includes votes by each member of the U.S. House of Representatives on 16 issues. It has 435 objects with the label 'Republican' or 'Democrat', of which 168 are Republican and 267 are Democrat. The 16 issues correspond to 16 attributes with the value of 'Yes', 'No', or '?' (for missing values or abstentions). The value '?' is treated the same as the other values, so the number of categorical values is J=48.

The *Mushroom* data set contains 8124 objects (3916 poisonous and 4208 edible), with 22 attributes. All the attributes are categorical, describing the physical characteristics (color, size, shape, etc.) of a single mushroom. The 'missing' value is treated as a separate categorical value, the same as the others, for each attribute, and the number of categorical values is J=117.

The Internet Advertisements data set (*Internet Ads*) contains 3279 objects representing a set of possible advertisements on Web pages. The original data have 1558 attributes, 3 continuous and 1555 Boolean. We use only the 1555 Boolean attributes. These attributes encode features from the url, origurl, ancurl, alt and caption terms. Each object is labeled either 'ad' or 'non-ad'. This is an unbalanced data set, as only 459 objects are advertisements and most are non-ads. The occasional missing values that occur are treated as '0', and thus the number of categorical values is J=3110. This data poses a greater challenge due to its high dimensionality and unbalanced character.

Tables 2.7, 2.8, 2.9 and 2.10 show the clustering results of DHCC on the four data sets, in the form of contingency tables, where the clusters are labeled using prefix code. These contingency tables demonstrate the great capability of DHCC to identify the natural structure. For the *Zoo* data, the mammal and non-mammal families are well separated at the first bisection, and the remaining two major families, i.e., bird and fish, are well distinguished at the second bisection. For the *Votes* data, DHCC generates two clusters that give a good separation between Republican and Democrat: Democratic votes, in particular, are well identified by cluster '1'. For the *Mushroom* data, we can see that the poisonous and edible mushrooms are readily distinguishable from the clustering results of DHCC. Even one split to the all-inclusive root yields two clusters that separate the two classes well, as the first level of the clustering tree has 3048 poisonous and 48 edible mushrooms in the left cluster, and 868 poisonous and 4160 edible in the right cluster. The primary objective on the *Internet Ads* data is to identify the Web advertisements, which account for only a small proportion of the whole set. DHCC generates 21 clusters on the *Internet Ads* data; Table 2.10 only presents the clustering tree up to the second level. From Table 2.10 we can see that DHCC generates three main clusters at the second level of bisection, i.e., the non-ads cluster '0', the ads cluster '11' and the ambiguous cluster '10'. The cluster '0', which contains the majority of the non-ads with a few ads, is further partitioned into 8 clusters. The cluster '11' effectively identifies the majority of the Web advertisements, grouping 268 ads and 69 non-ads, and this cluster is further partitioned into 12 clusters.

Cluster	Mammal	Bird	Reptile	Fish	Amphibian	Insect	Invertebrate
0	41	0	0	0	0	0	0
10	0	0	5	13	4	0	8
11	0	20	0	0	0	8	2

Table 2.7: Contingency table of the clustering results of DHCC on *Zoo*

Cluster	Republicans	Democrats
0	159	49
1	9	218

Table 2.8: Contingency table of the clustering results of DHCC on *Votes*

The comparison of DHCC with AT-DC, COOLCAT, SUBCAD and K-modes in terms of *NMI* is given in Figure 2.8; and the comparison by *CU* is given in Table 2.11. Note that CACTUS was unable to generate clusters on any of the four real-life data sets, so its results are not presented here. The *CU* values of AT-DC are not given in Table 2.11 because AT-DC generates different numbers of clusters from DHCC on these real-life data sets, and the comparison of clustering results for different K in terms of *CU* is biased. To compare objectively, we set the number of clusters for COOLCAT, SUBCAD and K-modes to both the number of leaf clusters and the minimum (optimum) number of clusters that can discover the natural classes by cutting the clustering tree generated by DHCC. The two values are the same on the *Zoo* and *Votes* data, while they are different on *Mushroom* and *Internet Ads*, the minimum (optimum) values for *Mushroom* and *Internet Ads* being 2 and 3 respectively. We also present the *NMI* values for the optimum number of clusters. As we did for the synthetic data, we ran AT-DC, COOLCAT, SUBCAD and K-modes 10 times with different initializations and processing orders, the 10 processing orders being the original order, the reversed original order, the order associated with class labels and seven random orders [8]. The best, worst and average results in terms of *NMI* and *CU* on each data set for each of these five algorithms are given in Figure 2.8 and Table 2.13 respectively, represented in the same format as for the synthetic data. Again, we use the entire set as a sample for COOLCAT and SUBCAD.

[8]As the objects in the *Internet Ads* data set were originally ordered in association with class labels, eight random orders were used.

Cluster	Poisonous		Edible	
000	24	3048	32	48
001	1728		0	
010	1296		0	
011	0		16	
10	736	868	2880	4160
110	72		808	
11100	0		216	
111010	44		0	
111011	16		64	
1111	0		192	

Table 2.9: Contingency table of the clustering results of DHCC on *Mushroom*

Cluster	Ads	Non-ads
0	49	1570
10	142	1181
11	268	69

Table 2.10: Contingency table of the clustering results of DHCC on *Internet Ads*

Comparing DHCC with the average values for *CU* (Table 2.11) and especially for *NMI* (Figure 2.8) we see that it performs better than the other four algorithms on the *Zoo* data set, and its results are comparable to those of the winner on *Votes* (K-modes), *Mushroom* (COOLCAT) and *Internet Ads* (AT-DC). Furthermore, DHCC outperforms all of the other algorithms by a large margin on *Mushroom* and *Internet Ads* when the evaluation is conducted based on the optimum number of clusters, which is 2 for *Mushroom* and 3 for *Internet Ads*. When the strategy to deal with order-sensitivity is used, AT-DC is much more stable than COOLCAT, SUBCAD and the K-modes, showing no variance on the *Mushroom* data set and little on *Votes* and *Internet Ads*. AT-DC tends to group most of the data in one cluster on the *Internet Ads* data set on all runs. For example, it places 230 ads and 2137 non-ads in one cluster in the best run (by *NMI* – see Figure 2.8): from the perspective of identifying Web advertisements, it missed the majority of the targets. In terms of order dependence, SUBCAD performs the worst of all the algorithms: its results show great variance on all four data sets, and the partitions from some runs on all four

	Zoo	*Votes*	*Mushroom*	*Internet Ads*
DHCC	**3.642**	2.92	6.073	6.784
COOLCAT	3.632	2.937	8.017	8.347
	2.612	1.042	6.133	7.4
	3.353	2.608	**7.383**	**7.72**
SUBCAD	3.39	2.905	6.473	5.086
	0.665	0.121	0.877	3.452
	1.561	2.485	4.495	4.334
K-modes	2.923	2.939	3.895	n/a
	1.773	2.938	1.781	
	2.304	**2.939**	2.615	

Table 2.11: Comparison of DHCC in terms of *CU* on real-life data

data sets are meaningless with respect to the natural classification, as can be seen from the contingency tables in Tables 2.12 and 2.13. COOLCAT is also sensitive to the processing order, although its results show less variance on these real-life data than SUBCAD. An 'improper' order can lead to very poor clustering, as can be seen in the worst results on the *Votes* data, shown in the form of a contingency table in Table 2.14. The K-modes is very sensitive to the initialization: an 'improper' initial assignment can lead to meaningless clustering, as can be seen in the worst cases on the *Mushroom* data. Also, in most cases, the K-modes converges to one cluster on the *Internet Ads set*; this is because the high dimensionality of the data renders the objects almost equidistant from each other under the distance measure used by the K-modes. Its results on the *Internet Ads* data set are therefore not presented. DHCC thus demonstrates superior performance in terms of both clustering quality and stability.

Cluster No.	Republicans	Democrats
1	9	49
2	159	218

Table 2.12: Contingency table of the worst clustering results of SUBCAD on *Votes*

Cluster No.	Poisonous	Edible
1	144	0
2	3609	3780
3	34	158
4	0	192
5	0	18
6	12	2
7	18	0
8	41	0
9	26	19
10	32	39

Table 2.13: Contingency table of the worst clustering results of SUBCAD on *Mushroom*

Cluster No.	Republicans	Democrats
1	160	158
2	8	109

Table 2.14: Contingency table of the worst clustering results of COOLCAT on *Votes*

2.8 Chapter summary

We have designed a new systematic, efficient and effective divisive hierarchical clustering algorithm for categorical data, named DHCC. The splitting procedure in DHCC consists of two phases, preliminary splitting and refinement. Preliminary splitting is carried out based on MCA, and the refinement phase optimizes the global objective function SCE locally to improve the quality of the bisection. Using a new data representation method and a Chi-square dissimilarity measure, DHCC is capable of seamlessly discovering clusters embedded in subspaces. DHCC is parameter-free, fully automatic, and independent of the order in which the data is processed. Additionally, its time complexity is linear with respect to the number of objects for clustering and the total number of categorical values (and thus also to the dimensionality). Experiments on both synthetic and real-life data show that DHCC yields high-quality results.

We have also devised a termination strategy based on a new clustering quality measure to generate the proper top part of the hierarchical clustering tree. This new

global quality measure can also be applied to an agglomerative hierarchical clustering algorithm for categorical data to decide the number of clusters; for example, the process of merging the most similar pair of clusters terminates when the quality of resulting clustering decrease

Finally, clustering extremely large data sets requires combination of sampling and clustering techniques. When a clustering algorithm is used on a sample data, there is naturally the object allocation problem. Using the definition of a cluster center for categorical data and the Chi-square distance between an individual categorical object and a cluster, the problem of allocating unclustered objects to properly pre-constructed clusters [17] could be effectively dealt with.

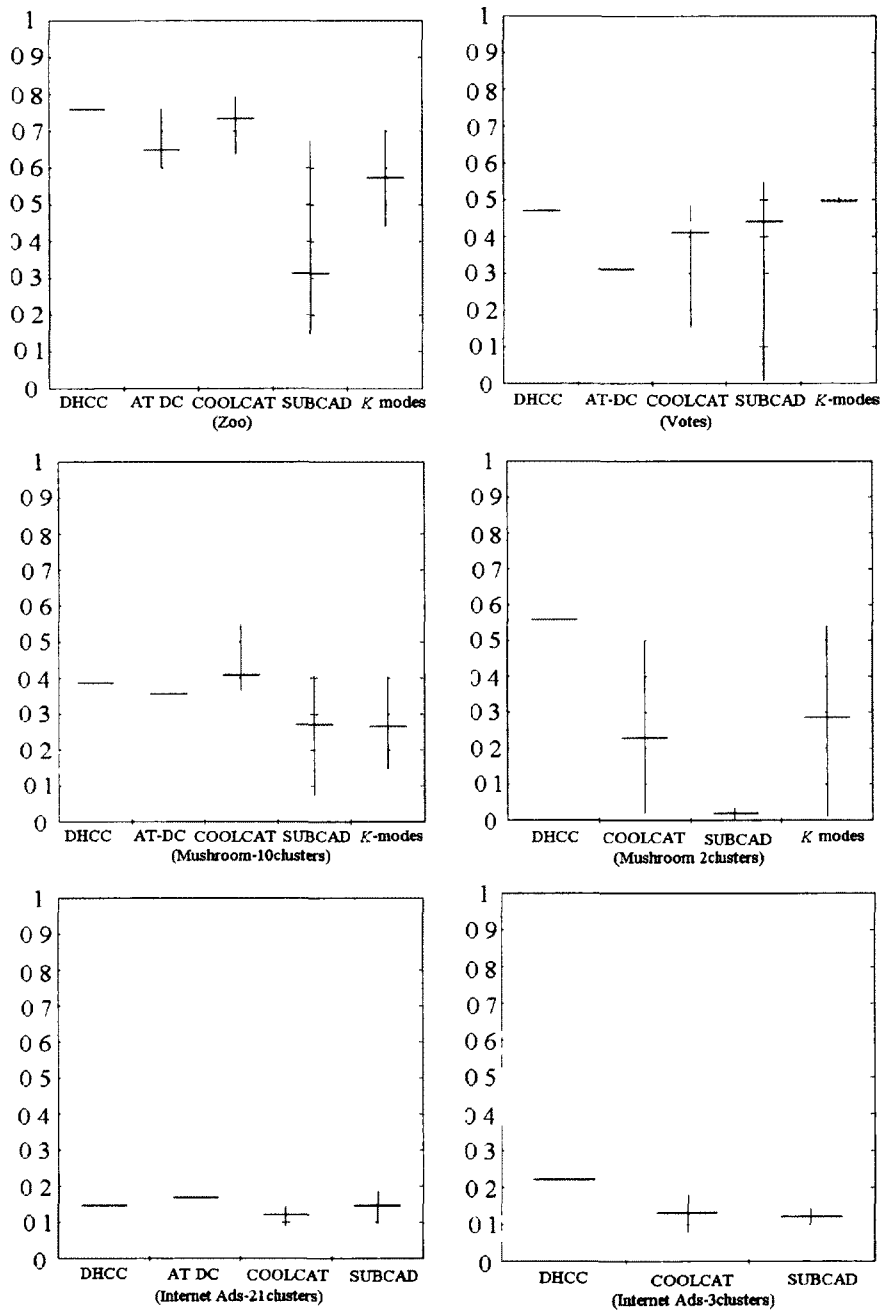

Figure 2.8: Comparison of DHCC in terms of *NMI* on real-life data

Chapter 3

Semi-supervised Clustering of Categorical Data

This chapter describes a novel semi-supervised divisive hierarchical algorithm for categorical data, named SDHCC [91], which is underlain by DHCC described in Chapter 2. We view semi-supervised clustering of categorical data as an optimization problem with extra instance-level constraints, and propose a systematic and fully automated approach to guide the optimization process to a better solution in terms of satisfying the constraints, which would also be beneficial to the unconstrained objects.

3.1 Introduction

Semi-supervised clustering can yield considerable improvement over unsupervised clustering [9, 11, 23, 51, 54, 55]. With large amounts of categorical data being generated in real-life application, clustering categorical data has been receiving increasing attention in recent years, as described in Chapter 2. Moreover, in many real-life applications on categorical data, prior knowledge is omnipresent in relation to the need or the goal of the data analysis. For example, when a market analyst performs clustering for customer segmentation from survey data, he/she will likely want that

the customers from the same family be grouped together instead of trivial separation between men and women. However, none of existing algorithms for clustering categorical data can exploit prior background knowledge. In this chapter, we investigate how to utilize the prior background knowledge of categorical data to guide the clustering process of DHCC.

We propose a semi-supervised clustering algorithm for categorical data. The new algorithm exploits the same basic Multiple Correspondence Analysis (MCA) technique used in DHCC, which is described in Chapter 2. DHCC is an unsupervised divisive hierarchical algorithm for clustering categorical data. The new algorithm proposed in this chapter also adopts the divisive hierarchical approach to clustering categorical data. The optimization problem in the clustering is NP hard [4], and in DHCC, the objective function SCE is locally minimized in each bisection step. In this chapter, we propose a systematic and fully automated approach to a better solution, using prior background knowledge. Under the new optimization framework, DHCC can be viewed as a special case with no instance-level constraint input. Our method does not need to introduce any parameters into the underlying algorithm, and the additional operations involved are independent of the processing order and have linear time complexity. We have named the new algorithm SDHCC (Semi-supervised Divisive Hierarchical Clustering of Categorical data). To our knowledge, SDHCC is the first semi-supervised clustering algorithm for categorical data.

3.2 Related work

The majority of the existing semi-supervised clustering algorithms are non-hierarchical and focus on analyzing numeric data [23]. Most existing non-hierarchical clustering algorithms are derived from the K-means algorithm [9, 22, 54, 78, 82]. These K-means variants use the background information either to guide the assignment of the objects [22, 82] or to weight the similarity measures, by penalizing constraint violation [9] or rewarding constraint satisfaction [54], or a hybrid of these two methods [78]. Besides the K-means variants, some algorithms are derived from density-based clustering algorithm DBSCAN [56, 70]. In the DBSCAN algorithm [31], appropriately setting the

parameters, a minimum number of points $MinPts \in \mathbb{N}$, and a radius $\epsilon \in \mathbb{R}$, is critical but difficult, especially when the density among the clusters differs widely. In these density-based semi-supervised clustering algorithms, labeled objects or instance-level constraints have been used to help set the parameters of the DBSCAN algorithm. However, neither the concept of objective function or Euclidean distance [1] used in the non-hierarchical methods nor the density notion in density-based clustering algorithms is naturally meaningful for categorical data.

Little work has been done on the application of background knowledge to hierarchical clustering [51, 23], and even these few published hierarchical algorithms are all agglomerative, not divisive. The concept of comparing similarity between pairwise objects, which is used in agglomerative method, is not suitable for categorical data [40, 89, 92], thus the error-propagation issue of agglomerative in categorical domain is even more critical. In [51], the shortest-path (Floyd-Warshall) algorithm [19] is exploited to propagate the constraints, and the complete-link agglomerative method is used to construct the clustering hierarchy; and thus the time complexity of which is $O(n^3)$ with respect to the number of object n. The algorithm in [23] is similar to traditional agglomerative hierarchical method except that it initially assembles the objects in the same must-link closure together, and its time complexity is $O(n^2)$. The high time complexity of agglomerative methods prevents them from being used on very large data sets.

A benchmark *search-based* algorithm, COP-KMEANS [82], is described in Figure 3.1. It was used as a competitor in clustering performance comparison in this chapter. In the implementation of the algorithm, the similarity measure we used is the one proposed in a K-modes algorithm for categorical data in (San et al. 2004); this measure is described in Section 2.2 in Chapter 2. The modified COP-KMEANS is called SKModes in this thesis.

All the constraints in COP-KMEANS are satisfied. When assigning an object X_i, it sorts all the clusters according to the similarity of X_i between them, and continues down the sorted list until finds one that can legally host X_i. The algorithm can reach a dead end, that is, it is unable to assign X_i as all the clusters have violation with it,

[1] or cosine similarity which is equivalent to Euclidean distance in normalized unit vectors

Algorithm: COP-KMEANS
Input: Data set T, *must-link* constraints $Con_{=}$, *cannot-link* constraints $Con_{\neq}$

1. Let $C_1, \cdots, C_K$ be the initial cluster centers.
2. For each object X_i in T, assign it to the closet cluster C_j such that VIOLATE-CONSTRAINTS(X_i, C_j, $Con_{=}$, $Con_{\neq}$) is false. If no such cluster exits, fail (return { }).
3. For each cluster C_j, update its center using all the objects that have been assigned to it.
4. Iterate step (2) and (3) until convergence.
5. Return $\{C_1, \cdots, C_K\}$

Figure 3.1: COP-KMEANS algorithm

even the number of clusters K is properly set.

3.3 Notation and definitions

The notation used in this chapter is consistent with that in Chapter 2. In this section, we define the extra notation used for instance-level constraint knowledge. In this thesis, prior background knowledge is provided as *must-link* and *cannot-link* constraints on pairs of objects [78, 82]. A *must-link* constraint indicates that the two objects have to be in the same cluster, while a *cannot-link* constraint indicates that the two objects must not be placed in the same cluster. Labeling objects sometimes is difficult as full class information is not available, whereas specifying whether two objects should or should not be placed together is more natural and practical. Moreover, pairwise object constraints are more general than labeled objects because the latter can be transformed into equivalent pairwise object constraints, but not the reverse.

Since *must-link* constraints are equivalence relations, they can be used to generate

transitive closures. *Cannot-link* constraints then can be transformed to a *cannot-link* matrix representing link relationship between closures. Figure 3.2 illustrates the generation of transitive closures from *must-link* constraints, where a solid line indicates that the two objects must be linked and a dotted line, that the two objects cannot be linked. For example, in Figure 3.2, object $a1$ must link to $a2$, and $a2$ must link to $a3$, then $a1$, $a2$ and $a3$ are in one transitive closure, indicating they have to be put in the same cluster. In this thesis, a transitive closure is also called a constraint closure. Besides the transitive closure, a single object which is not involved in any *must-link* constraint but is involved in a *cannot-link* constraint is also called a constraint closure. For example, $d1$ in Figure 3.2 is a singleton constraint closure.

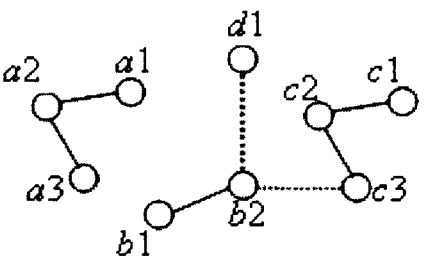

Figure 3.2: Constraint closures generated from instance-level constraints

The *cannot-link* matrix is generated from the constraint closures and the *cannot-link* constraints. Let l be the number of constraint closures; then the *cannot-link* matrix M is of order $l \times l$, where $m_{ij} = 1$ if closure c_i and c_j cannot link, otherwise, $m_{ij} = 0$. Closures c_i and c_j cannot link if $\exists X_i \in c_i$ and $\exists X_j \in c_j$ such that (X_i, X_j) is in the set of *cannot-link* constraints. Thus the *cannot-link* matrix M is a symmetric Boolean matrix.

The feasibility of satisfying all constraints for non-hierarchical and agglomerative hierarchical clustering has been studied in [22, 23]. In agglomerative hierarchical clustering, the operation of merging the two closest clusters terminates when a dead end, i.e., merging any pair of current clusters leads to the *cannot-link* violation, is reached [23]. However, the feasibility problem is more complex in divisive hierarchical clustering, because a divisive method starts with an all-inclusive cluster containing all the objects, and repeatedly chooses one cluster to split into two subclusters, which

may make violations of *cannot-link* constraints unavoidable lower down (closer to the root) in the clustering hierarchy. For example, when there are more than two classes which mutually cannot link in the data, violations of cannot-link constraints are inevitable at the lowest level of the clustering tree, if binary tree is used to represent the hierarchical structure of the clustering results (which is however commonly used in hierarchical clustering). In this thesis, *must-link* constraints are satisfied at all levels of the clustering tree, whereas *cannot-link* violation is tolerated, especially at the lower levels (closer to the root). Note that we assume the constraints are consistent, dealing with the erroneous constraints is out of the scope of this thesis. The following definitions help to measure the degree of *cannot-link* violation in a cluster (under the assumption that all the *must-link* constraints are satisfied). These definitions will be used in Section 3.4.

Definition 3.1. *Given two constraint closures c_i and c_j, and the cannot-link matrix M, the cannot-link weight of the closure c_i with respect to c_j is*

$$w_i(j) = \begin{cases} |c_j|, & m_{ij} = 1; \\ 0, & m_{ij} = 0. \end{cases}$$

where $|c_j|$ is the number of objects in closure c_j.

Definition 3.2. *The degree of cannot-link violation of cluster C is defined as:*

$$w(C) = \sum_{c_i \in C} w(c_i|C) = \sum_{c_i \in C} \sum_{c_j \in C} w_i(j) \tag{3.1}$$

where $w(c_i|C) = \sum_{c_j \in C} w_i(j)$ is the cannot-link weight of c_i in cluster C.

3.4 Semi-supervised DHCC

The overall scheme of SDHCC is the same as that of DHCC, which is presented in Chapter 2. The optimization problem of clustering categorical data defined in Chapter

2 is NP hard. DHCC attempts to find a sub-optimal (if not optimal) solution with K=2 for each bisection. In SDHCC, our proposed approach tries to use constraint knowledge to guide the optimization process to a better solution in terms of satisfying the constraints. Each bisection step in SDHCC consists of three phases: initialization of bisection; iterative refinement of the bisection based on Chi-square distance; and alleviation of the *cannot-link* violation. The framework of bisection procedure is shown in Figure 3.3. The detailed description of these three phases will be presented in the following subsections.

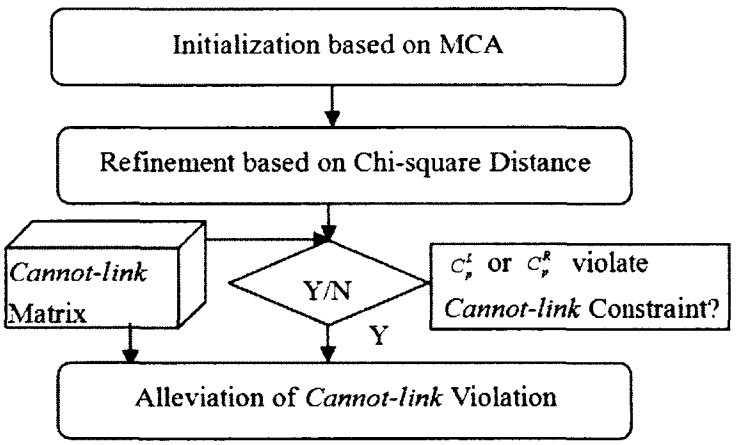

Figure 3.3: The framework of the bisection procedure of SDHCC

3.4.1 Initialization

In each step of bisection, the initialization of the bisection consists of two steps, i.e., preliminary splitting based on multiple correspondence analysis (MCA), and redistributing the objects involved in must-link constraints to satisfy the must-link constraints.

The first step of bisection initialization proceeds the same as the preliminary splitting in DHCC. To bisect a cluster C_p with $|C_p|$ objects, we apply MCA on the indicator matrix $Z^{(p)}$ of order $|C_p| \times J$ from the $|C_p|$ objects to get the left singular vectors $U^{(P)}$. Each object Z_i whose first coordinate $U_{i1}^{(p)} \leq 0$ goes to the left child of

C_p, and each Z_i whose first coordinate $U_{i1}^{(p)} > 0$ goes to the right child of C_p.

The second step aims to satisfy all the *must-link* constraints. In fact, the preliminary splitting in first step may distribute the objects in a constraint closure in the two subclusters. In the second step, violation of *must-link* constraints is eliminated by forcing the objects in a closure to be re-assigned to the same subcluster. Each closure c split by the first step is re-assigned to the subcluster that holds most of its members; i.e., if $|c \cap C_p^L| \geq |c \cap C_p^R|$, the closure c goes to the left child C_p^L, otherwise, the closure c goes to the right child C_p^R.

3.4.2 Refinement

In the refinement phase, we employ the Chi-square distance to measure the dissimilarity between a single categorical object, as well as a constraint closure, and a cluster of categorical objects. In fact, after re-assignment of the second step of the initialization phase, the objects from each closure are assembled together, so the objects in a closure will be treated as an entity in performing the refinement. Therefore, we define the dissimilarity between a closure and a cluster of categorical objects as follows.

Definition 3.3. *The dissimilarity between a constraint closure c and a cluster C_i is the Chi-square distance between them, i.e.,*

$$d_{Chi}(c, C_i) = \sum_j \frac{(\bar{c}_j - \mu_{ij})^2}{\mu_{ij}} \tag{3.2}$$

where c is in cluster C_i, and $\bar{c}_j (1 \leq j \leq J)$ is the j^{th} element of the center of closure c, i.e., $\bar{c}_j = \left(\frac{1}{|c|}\sum_{Z_i \in c} z_{ij}\right)^{1/2}$ [2]. μ_{ij} is the j^{th} element of the cluster center of cluster C_i. When c is a singleton closure, Formula (3.2) is the same as Formula (2.8).

Theorem. *In a refinement phase (not limited to bisection operation of hierarchical clustering), deciding the membership of a closure according to the Chi-square distance*

[2]Here we consider closure c as a set of objects and applied Formula (2.9) to compute the center of the closure.

defined in (3.2) is equivalent to making the decision according to the sum of the Chi-square distance of all the objects in the closure, which is defined in (2.8) in Chapter 2.

Proof. The Chi-square distance in (3.2) is rewritten as

$$d_{Chi}(c, C_i) = \sum_j \frac{(\bar{c}_j - \mu_{ij})^2}{\mu_{ij}} = \sum_j \left(\frac{\bar{c}_j^2}{\mu_{ij}} + \mu_{ij} - 2\bar{c}_j \right) \tag{3.3}$$

The value of $\sum_j \bar{c}_j$ is independent of the cluster center, which makes the same contribution in the calculation of the distance between closure c and the different cluster centers C_i, and can thus be omitted in the process of reassignment. So the membership of c is determined by

$$\sum_j \left(\frac{\bar{c}_j^2}{\mu_{ij}} + \mu_{ij} \right) = \sum_j \left(\left(\frac{1}{|c|} \sum_{Z_i \in c} z_{ij} \right) / \mu_{ij} + \mu_{ij} \right) = \frac{1}{|c|} \sum_{Z_i \in c} \sum_j \left(\frac{z_{ij}}{\mu_{ij}} + \mu_{ij} \right) \tag{3.4}$$

Analogously, derived from Formula (2.8), the membership of the single object Z_i is determined by

$$\sum_j \left(\frac{z_{ij}}{\mu_{ij}} + \mu_{ij} \right) \tag{3.5}$$

Formula (3.4) is the average of the values of Formula (3.5) over all the objects in closure c. Therefore, Formula (3.2) is equivalent to the sum of the distances of all the objects in closure c, in terms of deciding the membership of closure c.

Definition 3.3 and Theorem 3.1 provide a significant computational advantage. They show that it is only necessary to store the statistic features of each constraint closure and those of the two resulting subclusters, to speed up the refinement process. In other words, the objects in a closure is treated as an entity, the number of objects for refinement is thus reduced. The cluster features of C_p in SDHCC is the same as in DHCC, the features of a closure are the same as those of a cluster. We do not need to revisit all the objects in the cluster to update the cluster features; instead, we just need to record the objects which were relocated to do the update, so the cluster features can thus be updated efficiently after each iteration cycle.

The statistic feature of Chi-square distance requires that individual object (or closure) be in the measured set. As the Chi-square distance in (2.8) and (3.2) measures the association between an individual object/closure and its group, when calculating the distance between an object Z_i and cluster C_p, where Z_i is not in C_p, we should proceed as if Z_i is in cluster C_p by updating the cluster features z_j to $r_p \times z_j + z_{ij}$, $|C_p|$ to $r_p \times |C_p| + 1$, and after that, restore the cluster features to z_j and $|C_p|$. Here r_p is the balance ratio of cluster size of the two resulting subclusters. The balance ratio is set to 1.0 for C_p^L, and $|C_p^L|/|C_p^R|$ for C_p^R if $|C_p^L| \geq |C_p^R|$, vice versa if $|C_p^L| \leq |C_p^R|$. The same applies to the distance calculation between a closure c and C_p, where c is not in C_p. The re-assignment based on Chi-square distance has greater tendency to assign an individual object/closure to the smaller one of the two resulting subclusters. This is because that the rare categorical value of the individual object/closure has relative high frequency in smaller cluster, which leads to smaller Chi-square distance. Therefore, we use the balance ratio to solve the bias issue in the relocation process.

In SDHCC, the refinement of the objects in C_p^L and C_p^R after initialization proceeds as in the algorithm in Figure 3.4.

Algorithm: Refinement after initialization

1. Calculate the cluster features of C_p^L and C_p^R, i.e., o_p^L, $|C_p^L|$ and o_p^R, $|C_p^R|$.
2. For each object Z_i in C_p^L
   if $Z_i \in c$ {if $d_{Chi}(c, C_p^R) < d_{Chi}(c, C_p^L)$, move c to C_p^R;}
   else if $d_{Chi}(Z_i, C_p^R) < d_{Chi}(Z_i, C_p^L)$, move Z_i to C_p^R;
   For each object Z_i in C_p^R
   if $Z_i \in c$ {if $d_{Chi}(c, C_p^L) < d_{Chi}(c, C_p^R)$, move c to C_p^L;}
   else if $d_{Chi}(Z_i, C_p^L) < d_{Chi}(Z_i, C_p^R)$, move Z_i to C_p^L;
3. Update the cluster features of o_p^L, $|C_p^L|$ and o_p^R, $|C_p^R|$.
4. Repeat steps (2) and (3) until the membership no longer changes.

Figure 3.4: Algorithm of the second step of SDHCC

3.4.3 Alleviation of *cannot-link* violation

Finding a feasible solution for satisfying the *cannot-link* constraints is a more difficult problem than addressing the *must-link* constraints [22, 23, 78]. In divisive hierarchical clustering, it is inevitable to have *cannot-link* constraints violated at the low (closer to the root) levels of the clustering hierarchy. In this subsection, a novel divide-and-merge method is proposed to alleviate the *cannot-link* violation in each bisection. This process takes advantage of the informative constraint knowledge to guide the assignment of unconstrained objects.

This phase attempts to alleviate the *cannot-link* violation in the two subclusters C_p^L and C_p^R resulting from the division of cluster C_p. Our proposed divide-and-merge method proceeds as follows (The pseudo-code is given in the algorithm in Figure 3.5):

Dividing operation: in each subcluster (C_p^L or C_p^R), if it has a *cannot-link* constraint, divide it into two sets. One of these, called the *alien* set, contains the target constraint closure which is most dissimilar with the subcluster, and also the objects that are similar to the target closure; the other set, which is called the *native* set, contains the rest of the objects in the subcluster. The pseudo-code for the dividing operation is shown in the algorithm in Figure 3.6.

Merging operation: After dividing operation, the alien set is merged with the native set of the other subcluster if doing so can decrease the sum of the degree of *cannot-link* violation of C_p^L and C_p^R. The pseudo-code for the merging operation is shown in the lgorithm in Figure 3.7.

The function of the divide-and-merge operation is twofold: alleviating the *cannot-link* violation and guiding the assignment of unconstrained objects. The constraint closure which has the largest *cannot-link* weight in the subcluster is chosen as the target closure; if in a subcluster there is more than one closure with the largest *cannot-link* weight, choose the one with the greatest Chi-square distance from the cluster. To use the constraint knowledge to guide the assignment of unconstrained objects, the unstrained objects which are similar with the target closure are removed together. The divide-and-merge operation on C_p^L and C_p^R proceeds iteratively until the sum of

the degree of *cannot-link* violation of C_p^L and C_p^R cannot decrease. (The definitions of the degree of *cannot-link* violation of a cluster and the *cannot-link* weight of a closure in a cluster are given in Section 3.3.)

Algorithm: Alleviating the *cannot-link* violation

1. if C_p^L has *cannot-link* violation
   find target closure c_t^L;
   Divide-Cluster(C_p^L,c_t^L); //divide C_p^L into C_p^{LL} and C_p^{LR}, $c_t^L \subseteq C_p^{LR}$
   else $C_p^{LL} = C_p^L$; $C_p^{LR} = \emptyset$;

2. if C_p^R has *cannot-link* violation
   find target closure c_t^R;
   Divide-Cluster(C_p^R,c_t^R); //divide C_p^R into C_p^{RL} and C_p^{RR}, $c_t^R \subseteq C_p^{RR}$
   else $C_p^{RL} = C_p^R$; $C_p^{RR} = \emptyset$;

3. if($C_p^{LR} \neq \emptyset$ or $C_p^{RR} \neq \emptyset$)
   Merge-Clusters(C_p^{LL},C_p^{LR},C_p^{RL},C_p^{RR});

4. Repeat steps (1), (2) and (3) until no target closure can be found to decrease the sum of the degree of *cannot-link* violation of C_p^L and C_p^R.

5. Recall the algorithm in Figure 3.3.

Figure 3.5: Algorithm of the third step of SDHCC

We make use of the global clustering quality measure (2.12) proposed in Chapter 2 to decide when to terminate splitting in the algorithm in Figure 3.6 and the splitting process in constructing the clustering tree. The termination condition is that either of the two conditions given below is satisfied:

1. The algorithm in Figure 3.4 ends with one cluster, which means that the relocation algorithm finally converges into placing all the objects into one cluster.

2. Clustering quality does not increase, i.e.,$Q(\{C^t\}) \geq Q(\{C^{tL}, C^{tR}\})$, and neither C^{tL} nor C^{tR} violates the cannot-link constraints.

Algorithm: Divide-Cluster
Input: Cluster C, target closure c_t, $c_t \subset C$
Output: Cluster C^L, C^R, $C^L \cup C^R = C$; $C^L \cap C^R = \emptyset$; $c_t \subseteq C^R$

1. Initialize cluster $C^t = C$
2. Repeat until termination condition is satisfied
   (a) Initialize bisection of C^t;
   (b) Call the algorithm in Figure 3.3 to refine the objects in C^{tL} and C^{tR};
   (c) if $(c_t \subseteq C^{tL})$, $C^t = C^{tL}$; else $C^t = C^{tR}$;
3. $C^R = C^t$; $C^L = C - C^R$

Figure 3.6: Algorithm of Divide-Cluster

Algorithm: Merge-Clusters
Input: Clusters C_p^{LL},C_p^{LR},C_p^{RL},C_p^{RR}
Output: Cluster C_p^L, C_p^R
if $w(C_p^{LL} \cup C_p^{RR}) + w(C_p^{RL} \cup C_p^{LR}) < w(C_p^L) + w(C_p^R)$
  $C_p^L = C_p^{LL} \cup C_p^{RR}$; $C_p^R = C_p^{RL} \cup C_p^{LR}$.
else if $w(C_p^L \cup C_p^{RR}) + w(C_p^{RL}) < w(C_p^L) + w(C_p^R)$
  $C_p^L = C_p^L \cup C_p^{RR}$; $C_p^R = C_p^{RL}$.
else if $w(C_p^R \cup C_p^{LR}) + w(C_p^{LL}) < w(C_p^L) + w(C_p^R)$
  $C_p^L = C_p^{LL}$; $C_p^R = C_p^R \cup C_p^{LR}$.

Figure 3.7: Algorithm of Merge-Clusters

3.5 Experimental results

In this section, we study the performance of SDHCC on real data sets from the UCI Machine Learning Repository and 20-Newsgroup text data. As there is no published algorithm for semi-supervised clustering of categorical data, a straightforward and reasonable idea is to use existing K-modes algorithms for categorical data and K-means variants for semi-supervised clustering, as both of these kinds of algorithms extend the traditional K-means to broader applications. We implemented a semi-supervised clustering algorithm for categorical data by combining the semi-supervised

algorithm COP-KMEANS in [82] and the K-modes algorithm for categorical data in [72]; the combination is called SKModes in this thesis, as mentioned in Chapter 3.2. We also compared our algorithm with SKModes, as well as the state-of-the-art semi-supervised clustering algorithms for numeric data, which are constrained agglomerative hierarchical clustering algorithm in [23], named AggHie in this thesis, and the algorithm based on Hidden Markov Random Fields [9], named HMRF-Kmeans in this thesis. Both AggHie and HMRF-Kmeans run on the indicator matrix of categorical data based on cosine similarity measure. We investigate the effectiveness of using instance-level constraint knowledge by comparing the clustering results of the semi-supervised algorithms with those of its underlying unsupervised algorithms.

3.5.1 Evaluation measure and methodology

In our performance evaluation, we adopt the F-measure as the clustering validation measure. The F-measure of cluster i with respect to class j is $F(i,j) = 2 \times precision(i,j) \times recall(i,j)/(precision(i,j) + recall(i,j))$, where the precision of cluster i with respect to class j is $precision(i,j) = u_{ij}/|C_i|$; and the recall of cluster i with respect to class j is $recall(i,j) = u_{ij}/u_j$, where u_{ij} is the number of objects in cluster C_i belonging to class j and u_j is the number of objects in class j. The F-measure of the clustering is defined as

$$F = \sum_j \frac{u_j}{n} \max_i F(i,j) \tag{3.6}$$

In hierarchical clustering, the maximum is taken over all clusters i at all levels; in partitioning clustering, the maximum is taken over the K clusters. The F-measure of cluster i with respect to class j evaluates the extent to which a cluster contains only objects of a particular class and all objects of that class. The F-measure of a clustering evaluates the degree to which the clustering structure exactly matches the external classification.

To evaluate the performance of the algorithms objectively, we used twofold cross-validation on each data set for 20 trials. In each trial, we randomly selected 50% of

the objects as the test set, and the F-measure was calculated only on the test set. The remaining half of the data set was used as a training set, and the constraints were generated by randomly selecting pairs of objects from the training set, creating a *must-link* constraint if the pair objects have the same label and a *cannot-link* constraint if they have different labels. The clustering algorithms were run on the whole data set, and the results given are the averages of the results of the 20 trials.

The following variants of the algorithms are compared. For comparison, unsupervised variants were also evaluated on the test set only in each trial.

- SDHCC-M-C is the complete SDHCC algorithm with all three phases described in Section 3.4, i.e., initialization, refinement based on Chi-square distance (M) and alleviation of the *cannot-link* violation (C);
- SDHCC-M is an ablated version of SDHCC algorithm with only two phases, i.e., initialization and refinement based on Chi-square distance. The *cannot-link* constraints are only used in the termination condition to justify whether a cluster should be split or not;
- Unsupervised algorithm. When the number of constraints is zero, the results are from the underlying unsupervised clustering algorithms.

3.5.2 Results and discussion

UCI data set

Four data sets from the UCI Machine Learning Repository are used in our experiment to make performance evaluation. The four data sets are Zoo, Congressional Voting Records and Mushroom data sets which were used for performance evaluation in Chapter 2, plus another data set Wisconsin Breast Cancer. The description of the former three data sets is presented in Chapter 2, and the description of the Cancers data set follows.

The Wisconsin Breast Cancer data set (*Cancers*) has a total of 699 objects, 458 benign and 241 malignant, each of which is described by 9 categorical attributes. Each

attribute has 10 categorical values, and one attribute has a missing value, which is treated the same as the other values. Thus the number of categorical values is J=91.

Figures 3.8, 3.9, 3.10 and 3.11 show the clustering results on the *Zoo*, *Votes*, *Cancers*, and *Mushroom* data sets respectively. We set the number of clusters for SKModes to 7 on the *Zoo* data set. For *Votes* and *Cancers*, it was set to 3 for SKModes, as the algorithm fails to generate a clustering when the number of constraints is greater than 100 if we set the number of clusters to 2. The number of clusters was set to 4 for the *Mushroom* data set, and despite this, SKModes still fails to generate a clustering when the number of constraints is greater than 5000 in almost all of the 20 trials; the number of clusters for HMRF-Kmeans was set to the number of classes on all the data sets.

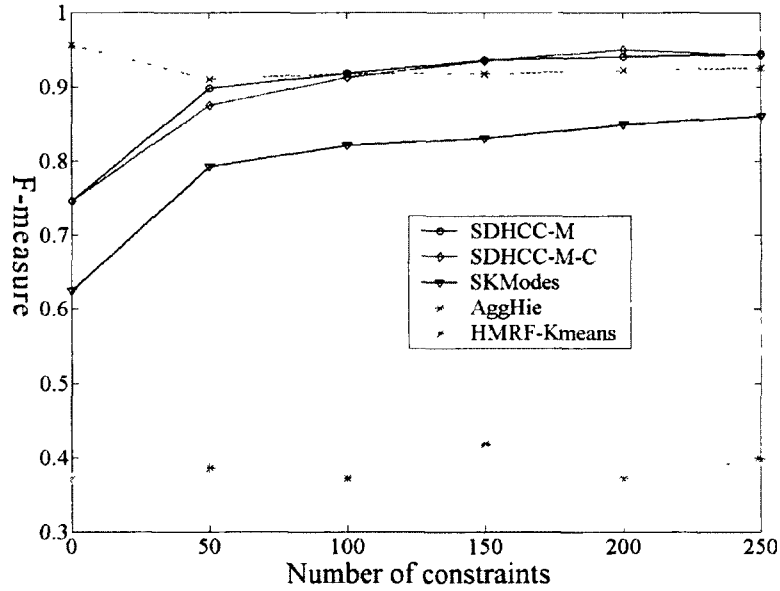

Figure 3.8: Semi-supervised clustering results on *Zoo* data set

From these figures we can see that the use of instance-level prior knowledge produces a remarkable improvement in SDHCC, except on the *Cancers* data set. On that set, SDHCC-M-C and SDHCC-M demonstrate the same clustering performance, and show trivial improvement compared with the result with no constraint involved. This is because unsupervised algorithm already yields outstanding clustering results, with

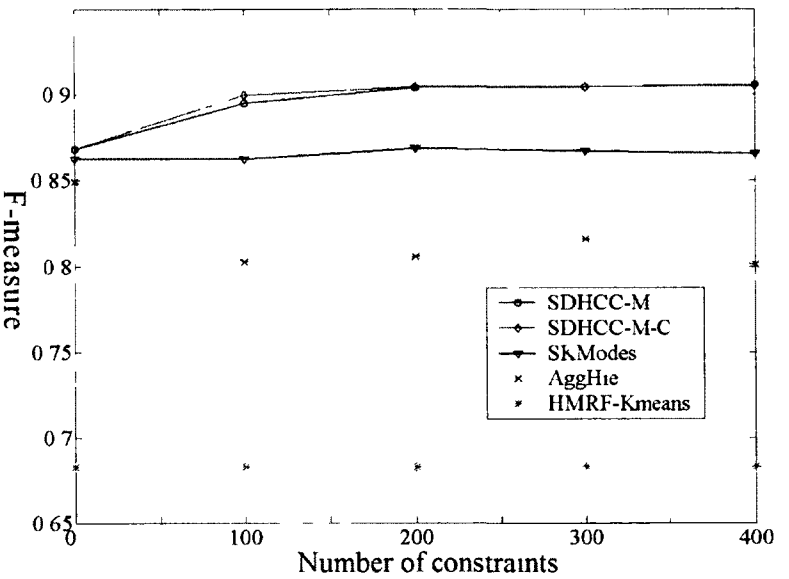

Figure 3.9: Semi-supervised clustering results on *Votes* data set

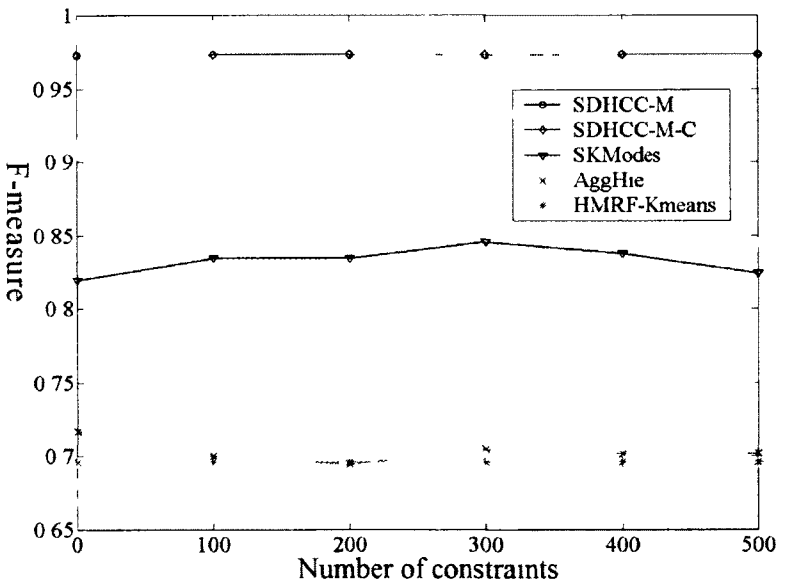

Figure 3.10: Semi-supervised clustering results on *Cancers* data set

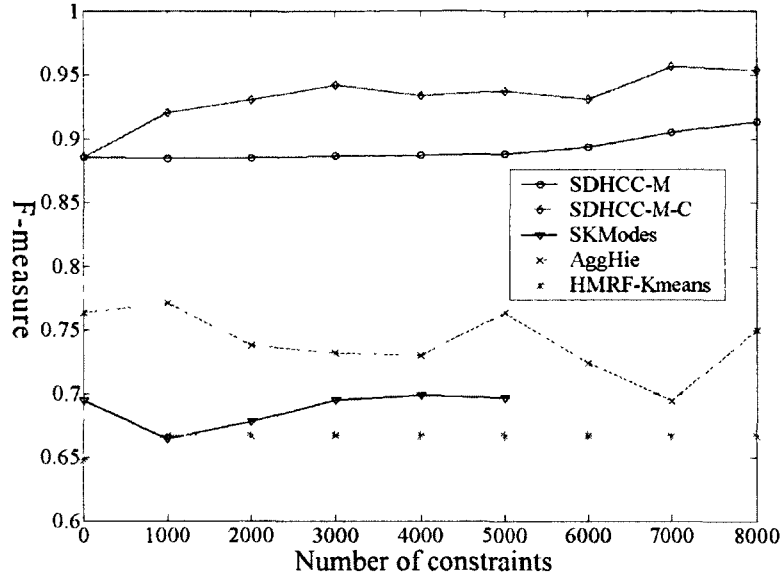

Figure 3.11: Semi-supervised clustering results on *Mushroom* data set

an F-measure attaining 0.97; thus, the instance-level constraints do not provide much informative knowledge to guide the clustering process. On the *Zoo* and *Votes* data sets, SDHCC-M-C and SDHCC-M show quite comparable clustering performance, while on the *Mushroom* data set, which is of higher dimension and much larger size, SDHCC-M-C outperforms SDHCC-M by a wide margin. For other three algorithms, none of them can reap potential benefit of prior knowledge on the four UCI data sets, except SKModes on the simple *Zoo* data set. Especially for AggHie, the clustering performance deteriorates when the instance-level constraints are incorporated. For HMRF-Kmeans, the other semi-supervised clustering algorithm for numeric data, the prior instance-level knowledge has little influence on the clustering performance, the clustering results show little variance with the increase of constraint knowledge on the four categorical data sets.

Overall, our complete SDHCC algorithm shows superior clustering performance on the UCI real-life data. On the *Cancers* data set, the value of F-measure is greater than 0.97, on *Zoo* and *Mushroom* data sets, the value reaches almost 0.96, and the value is also greater than 0.9 on the *Votes* data set, which means the clusters discovered by SDHCC-M-C closely match the natural classification.

20-Newsgroup text data

We also evaluate our algorithm and compare it with the three other algorithms on two different data sets extracted from 20-Newsgroup [3]. This text data is collected from 20 different Usenet newsgroups, each group consists of 1000 articles. We extract two data sets from the 20-Newsgroup data, and both data sets are preprocessed by removing stop-words and very high-frequency and low-frequency words, the same as the methodology used in [25]. After preprocessing, each data set is prepared with two versions, one is numeric data using TF-IDF weighting method; the other is categorical (transactional) data, where each article is simply represented by collection of the words appearing in the article (after preprocessing). The categorical data is very succinct representation of the articles, which eliminates the information of the frequency of each word. The description of the two data sets is as follows.

Similar-2: this data set consists of 200 articles from two similar topics, i.e., comp.sys.ibm.pc.hardware and comp.sys.mac.hardware, and each topic has 100 articles. The data is represented in 1233 dimensions (words), and in the categorical version, $J = 1233 \times 2$ as each word is analogous to one categorical attribute which takes two values indicating inclusion or non-inclusion of the word.

Different-3: this data set consists of 300 articles from three different topics, i.e., alt.atheism, comp.sys.ibm.pc.hardware and talk.politics.mideast, and each topic has 100 articles. The data is represented in 2470 dimensions (words), thus $J = 2470 \times 2$ in the categorical version.

Figures 3.12 and 3.13 show the clustering results on *Similar-2* and *Different-3* data sets respectively. For AggHie and HMRF-Kmeans, which are for numeric data, we give the clustering results on both numeric and categorical data presentation. We set the number of clusters to 2 on *Similar-2* and 3 on *Different-3* for SKModes and HMRF-Kmeans. In the figures, AggHie-N and HMRF-Kmeans-N indicate the results on numeric data, while AggHie-C and HMRF-Kmeans-C indicate the results on categorical data.

[3] 20-Newsgroup data. Available: `http://www.ai.mit.edu/people/jrennie/20Newsgroups`

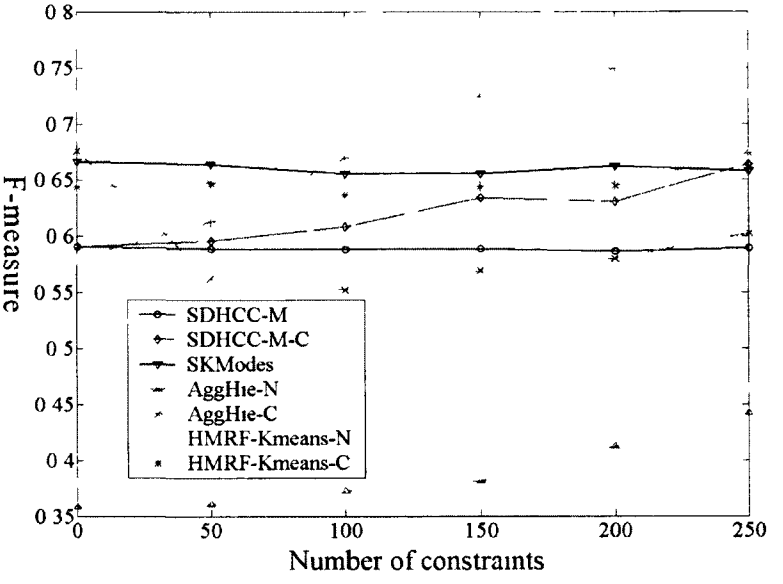

Figure 3.12: Semi-supervised clustering results on *Similar-2* data set

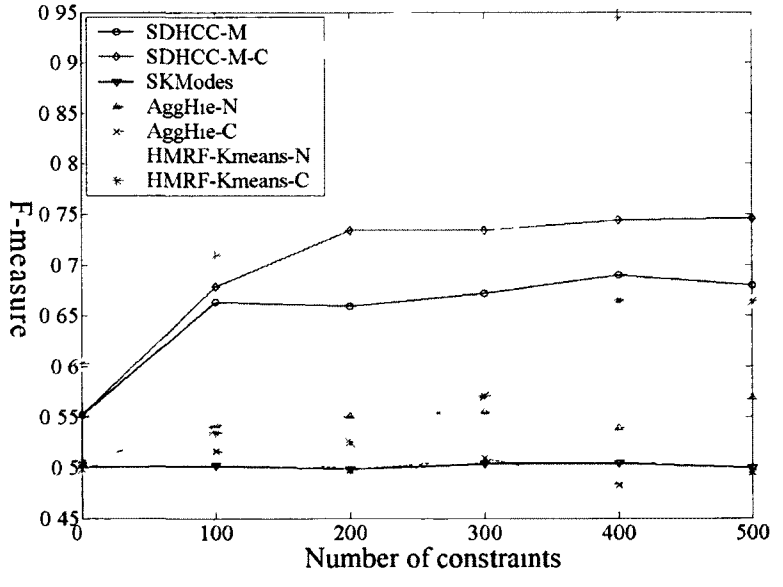

Figure 3.13: Semi-supervised clustering results on *different-3* data set

From these figures we can see that our algorithm can reap remarkable improvement even on the succinct categorical representation of text data, especially on the *Different-3* data set. *Similar-2* data impose more challenge than *Different-3* on clustering algorithms, since the overlap between the two topics in *Simila-2* is significant. SDHCC-M-C outperforms its ablated version SDHCC-M on the text data. HMRF-Kmeans demonstrates superior performance on numeric TF-IDF data; however, it gains much less improvement on the categorical data. SDHCC-M-C outperforms HMRF-Kmeans by a wide margin on categorical version of *Different-3* data set, comparably on *Similar-2* data. On the categorical *Similar-2* data, SDHCC-M-C performs best in terms of the improvement of clustering quality. AggHie reaps a little benefit from prior knowledge on numeric TF-IDF data, however, its performance on categorical version deteriorates when the instance-level constraints are incorporated, the same as it does on the UCI data set. AggHie-N has much smaller F-measure on *Similar-2* because it prematurely terminates with many clusters cannot be merged any more, as the pairwise similarity between the remaining clusters is zero measured by complete-link methodology. AggHie-N terminates more early than AggHie-C because the dimensionality of categorical data is twice as that of numeric data, the pairwise objects have more chance to show somewhat similarity. For SKModes, the prior instance-level knowledge has little influence on the clustering performance; the clustering results show little variance with constraint knowledge incorporated on the two text data sets.

Further discussion

As the results of both UCI data and text data demonstrate, SDHCC-M-C gains in performance very quickly at small numbers of constraints, and the improvement levels off with further increase in the number of constraints. This characteristic is very beneficial in real-life application domains, where the instance-level background knowledge would be provided by human experts, and it is very expensive to provide large numbers of pairwise constraints. Our semi-supervised clustering algorithm could reap the potential improvement afforded by a small amount of knowledge, which is very useful in real applications.

3.6 Chapter summary

In this chapter, we have proposed a semi-supervised divisive hierarchical clustering algorithm for categorical data (SDHCC). SDHCC builds on the novel MCA-based divisive hierarchical clustering algorithm for categorical data (DHCC). We exploit pairwise *must-link* and *cannot-link* constraint knowledge to guide the optimization process in SDHCC to a better solution in terms of satisfying the constraints, which would also be beneficial to the unconstrained objects. Experimental results on UCI data sets and 20-Newsgroup text data demonstrate that our semi-supervised algorithm shows remarkable improvement over the unsupervised clustering algorithm, yielding superior clustering results closely matching the natural classification on UCI data set, and remarkable improvement on the text data. Most importantly, our semi-supervised clustering algorithm could take advantage of the potential improvement from a small amount of knowledge, which is very useful in real applications. To our knowledge, SDHCC is the first semi-supervised clustering algorithm to deal with categorical data.

Our experiments also show that the mainstream semi-supervised clustering algorithms for numeric data, such as AggHie and HMRF-Kmeans, do not work well. Incorporating the instance-level constraint knowledge even could deteriorate the clustering quality of the underlying unsupervised clustering algorithm.

Chapter 4

Clustering Sequence Data

This chapter presents a statistical model for measuring the similarity between an individual categorical sequence and a set of categorical sequences. Based on the model, a novel model-based K-means algorithm is designed for clustering categorical sequences [88]. The comparison between a single sequence and a set of sequences based on the statistical model is alignment free, which can be applied to analyzing the sequences with varying length, and also the sequences whose significant structural features lie in different local places.

4.1 Introduction

Automatically analyzing sequence data has become increasingly important, with the upsurge in the amount of sequence data in the past few years, especially in biology area, such as genomic DNA sequences and unfolded protein sequences. Clustering technique is a powerful exploratory analysis tool, which can potentially reveal unknown categories of a given object, such as discovery of the unknown function of a protein, leading to better understanding of the nature of the sequence.

We circumvent the obstacle of defining a naturally meaningful pairwise similarity, by defining a similarity between an individual sequence and a cluster of sequences. Base on the new similarity measure defined on the conditional probability distribution (CPD) model [10, 93, 69], we design a novel model-based K-means clustering

algorithm for sequence clustering, which works in a similar way to the traditional K-means on vectorial data. Clustering sequence data is more challenging than clustering numeric data because, like clustering categorical data, there is no inherently meaningful measure of the similarity between categorical sequences. Usually, pairwise similarity measure is effective if there are significantly informative patterns in the sequences. However, it is difficult to define a meaningful pairwise similarity measure if sequences are short and contain noise, like the behavior sequences constructed from our project on personal bankruptcy prediction in Chapter 5.

4.2 Related work

Due to the lack of inherently meaningful measure of similarity for sequence, some measures, between pairwise sequences or between an individual sequence and a set of sequences, were proposed for sequence clustering and classification.

The nearest neighbor technique based on *edit distance* is one of the preferred methods for sequence clustering [2, 24, 93, 49, 63]. The definition of *edit distance* is given in Section 1.3 of Chapter 1. The major weakness of *edit distance* is that it only captures the optimal global alignment, but ignores the local similar structures hiding in the sequence. These significant local features present in considerably different positions in different order in the sequences, which make the sequences globally non-alignable. In many applications, the significant similarity between sequences lies in the local composition structure. Such as the protein sequences belonging to a broad family may have many similar local compositions but vary considerably from the view of global alignment. The block operations can alleviate the weakness of edit distance to certain extent [58, 63], however, edit distance with block operation is computationally intractable, and furthermore, it still fail to accurately measure the similarity through global alignment due to the different occurrences the local structure in different sequences.

Another approach widely used to measure the similarity between sequences is based on q-gram. The q-gram is a 'word' of length q, which is a subsequence of the original sequence. One such similarity measure is based on *Dice* coefficient, from

which, the similarity between sequence X and Y is defined as follows:

$$\frac{2 \times |q\text{-gram}(X) \cap q\text{-gram}(Y)|}{|q\text{-gram}(X)| + |q\text{-gram}(Y)|}$$

where q-gram(X) is the set of q-garms of sequence X. *Dice* coefficient with bigrams is a particularly popular sequence similarity measure [53]. The q-gram based approach extracts the 'words' and measures the similarity based on set calculation, taking no consideration of the order of the 'words' or their relative positioning in the sequence, which causes the major problem of this kind of similarity measure. For example, both 'xanex' and 'nexan' are composed of the same set of bigram, i.e., {xa, an, ne, ex}, however, these two sequences are radically different. Another practical issue of q-gram based approach is how to choose the proper length of the gram, i.e., q, in real-life application.

Statistical models are widely used for analyzing sequence data (in the categorical domain). Most statistical models of sequences are based on the hypothesis that occurrence of a symbol is closely related to what the preceding subsequence of length L is. The length of the subsequence L is referred to as memory length. The memory length is also called order in Markov chains or Hidden Markov Models (HMM). Because of the high complexity and inefficiency of Markov models [69, 74, 93], a statistical model based on the conditional probability distribution (CPD) of the next symbol given a preceding subsequence was proposed in [69], and applied successfully to the correction of corrupted text and DNA sequence classification. Given a subsequence σ, the model presents a conditional probability distribution over the finite alphabet Σ composing the sequences. The CPD is represented as $P(s_{s\in\Sigma}|\sigma)$, which denotes the conditional probability distribution of occurrence of the next symbol over Σ given the preceding subsequence σ.

The CPD model can be used to calculate the similarity between a single sequence, denoted by S, and a set of sequences, denoted by Δ [93]. Given the CPD, the probability of generating $S = s_1 s_2 \cdots s_l$ by the model can be calculated as:

$$P(S, \Delta) = P_\Delta(s_1) \times P_\Delta(s_2|s_1) \times \cdots \times P_\Delta(s_l|s_1 \cdots s_{l-1}) \tag{4.1}$$

where $P_\Delta(s_1)$ is the probability of presence of symbol s_1 out of all the sequences of Δ, which can be calculated by n_{s_1}/N_Δ , where n_{s_1} represents the times of occurrence of s_1, N_Δ represents the total number of symbols in the sequences of Δ, namely, the sum of the lengths of all the sequences of Δ; and $P_\Delta(s_i|s_1 \cdots s_{i-1})$ is the conditional probability that the symbol s_i right succeeds the subsequence $s_1 s_2 \cdots s_{i-1}$, which can be calculated by $n_{s_1 \cdots s_{i-1} s_i}/n_{s_1 \cdots s_{i-1}}$, where $n_{s_1 \cdots s_{i-1} s_i}$ and $n_{s_1 \cdots s_{i-1}}$ denote the times of occurrence of subsequence $s_1 \cdots s_{i-1} s_i$ and $s_1 \cdots s_{i-1}$, respectively, in Δ. The probability of generating a sequence S by a CPD model, which represents a cluster Δ, can be used as the similarity between S and Δ, that is, the higher the probability, the more likely S belongs to Δ.

Based on the CPD model, the CLUSEQ algorithm [93] was designed to cluster sequences into a set of possibly overlapped clusters. A set of sequences, denoted as Δ, is defined as a sequence cluster, if for each sequence S in Δ, the similarity between S and Δ is greater than or equal to some threshold t. CLUSEQ iteratively generate new clusters (refer to as new cluster generation) and dismiss (small) clusters that are covered by others (refer to as cluster consolidation), until the clustering does not change any more. Each new cluster at its initial stage contains only one sequence and is represented by the CPD model. To generate k new clusters, a set of k unclustered sequences need to be chosen as the seeds. At the first iteration, all sequences are unclustered. The flowchat of the CLUSEQ algorithm is given in Figure 4.1

4.3 A new measure of similarity between a categorical sequence and a cluster

A new similarity measure is proposed here to make the CPD model robust to noise situations. The new measure of similarity between an individual sequence S and a

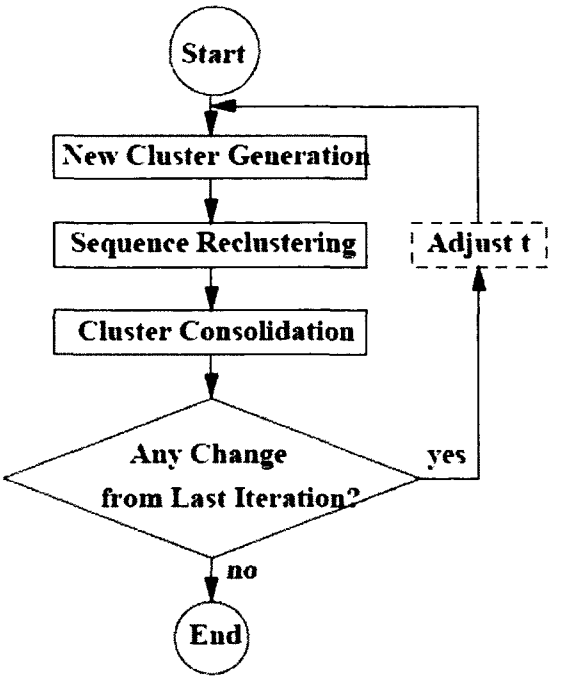

Figure 4.1: Flowchat of the CLUSEQ algorithm

cluster of sequences Δ is defined as follows:

$$\begin{aligned} sim_{\Delta}(S) &= \prod_{i=1}^{l} \exp\left(P_{\Delta}(s_i|s_1 \cdots s_{i-1}) - P(s_i)\right) \\ &= \frac{\exp\left(\sum_{i=1}^{l} P_{\Delta}(s_i|s_1 \cdots s_{i-1})\right)}{\exp\left(\sum_{i=1}^{l} P(s_i)\right)} \end{aligned} \tag{4.2}$$

where $P(s_i)$ is the probability of presence of symbol s_i out of the total set of sequences for clustering. So $P(S) = \prod_{i=1}^{l} P(s_i)$ is the probability of generating S out of total sequences based on memoryless occurrence frequency of the symbols. A $sim_{\Delta}(S) > 1$ indicates that the probability of generating S out of Δ based on CPD outweighs the probability of generating S out of whole sequences based on memoryless construction, which in turn indicates a high possibility that S belongs to Δ. The greater $sim_{\Delta}(S)$ is, the higher the possibility that S belongs to Δ.

In Formula (4.2), if $P_\Delta(s_i|s_1 \cdots s_{i-1}) > P(s_i)$, then $\exp(P_\Delta(s_i|s_1 \cdots s_{i-1}) - P(s_i)) > 1$, which means the symbol s_i contributes to increase $sim_\Delta(S)$. Otherwise, $sim_\Delta(S)$ is decreased by the factor $\exp(P_\Delta(s_i|s_1 \cdots s_{i-1}) - P(s_i))$, which is smaller than 1.

The main purpose to propose the improved new measure of similarity between a single sequence and a cluster is to lessen the influence of noise symbols in the sequences. There are generally two kinds of noise in the sequence classification domain. One is global noise symbols, i.e., the occurrence of the symbols is very rare from the global point of view. The other one is local noise symbols, i.e. the occurrence of the symbols is very rare within a cluster or their properties, if not significant, do not correspond to the cluster, while, except for the symbol(s) with quite few occurrences, the sequences show great similarity to the cluster. In the behavior sequences of our application of personal bankruptcy prediction, a bad account's occasional good behavior or a good account's occasional bad behavior can be treated as noise. Formula (4.1) suffers from the presence of noise. In fact, calculating similarity by chain multiplication of the probability of generating each symbol from a sequence cluster, the similarity value is sensitive to every symbol in the sequence S. If sequence S contains noise, which can be global noise or local noise, the probability of generating the noise symbols is very small (even equal to zero if the noise symbol does not appear in the sequences from the cluster), and these small noise probabilities will yield a very low similarity, no matter how similar S is to the cluster.

4.4 Calculation of the new measure

This section presents an efficient method for calculating the similarity proposed in Section 4.3 using the probability suffix tree (PST).

Given a sequence $S = s_1 s_2 \cdots s_l$, the preceding subsequence for calculating the conditional probability of generating s_i out of a cluster Δ should not always be $s_1 \cdots s_{i-1}$. In Formula (4.1) or (4.2), fixing the preceding subsequence of s_i as $s_1 \cdots s_{i-1}$ makes the assumption that all the subsequences are statistically significant. As a statistical model, if the number of occurrences $s_1 \cdots s_{i-1}$ is very small over Δ, the CPD of $P_\Delta(s_{s\in\Sigma}|s_1 \cdots s_{i-1})$ is statistically meaningless. To calculate CPD over

statistically significant subsequences, we use a significance threshold c to segment the preceding subsequence. A subsequence σ is significant if σ appears at least c times over Δ; otherwise it is insignificant. We define $s_j \cdots s_{i-1} (j \leq i)$ as a critical prefix of s_i if $s_j \cdots s_{i-1}$ is significant and $s_{j-1} \cdots s_{i-1}$ is insignificant. $j = i$ denotes the case in which there are very few occurrences of symbol s_{i-1} in Δ (maybe s_{i-1} is a noise symbol in Δ); in this case, we calculate $P_\Delta(s_i|s_1 \cdots s_{i-1}) = P_\Delta(s_i)$, where $P_\Delta(s_i)$ is the probability of presence of symbol s_i out of all the sequences of Δ. We choose the critical prefix of s_i to calculate the probability of generating s_i out of Δ, i.e., $P_\Delta(s_i|s_1 \cdots s_{i-1}) = P_\Delta(s_i|s_j \cdots s_{i-1})$. Therefore, the memory length for generating s_i is not fixed as $i - 1$, but varies depending on the statistical structure of the CPD model of Δ.

In the literature, the probabilistic suffix tree is employed to organize and implement the CPD model. The PST was originally introduced in [69], and is widely used in CPD model-based applications [10, 69, 93]. To retrieve the conditional probability of the critical prefix, we need to trace the preceding subsequence backward until the critical prefix is reached. So the suffixes of the prefixes of a set of sequences should be well organized for efficient retrieval. The prefixes of sequence $s_1 s_2 \cdots s_i$ are $s_1 \cdots s_j (j = 1, \cdots, i)$ and the suffixes of $s_1 s_2 \cdots s_i$ are $s_j \cdots s_i (j = 1, \cdots, i)$. For example, the prefixes of '1011' are '1', '10', '101' and '1011' and the suffixes of the prefixes are '1', '0', '10', '01', '101', '11', '011', and '1011'. A suffix tree is a rooted directed tree where each node is labeled by the suffixes of the prefixes of the sequences, accompanied by the number of occurrence of the label subsequence and CPD vector. The root node has no label, recording the total number of symbols in the cluster and the proportion of each symbol. These properties make the PST a desirable structure to organize the CPD model. An example of a PST is shown in Figure 4.2 [1]. Traveling the PST from the root to an internal node or leaf node, we get a reversed suffix, which serves as the label of the node.

[1]Figure 4.2 shows the illustration of a PST in ideal situations: that the occurrence frequency number of a node is equal to the sum of the numbers of its children. In real situations, the number is greater than the sum, due to the corresponding suffix of the node existing as the tail of the sequence, especially when there are many sequences and the sequences are short. In the ideal situation, the sequences constructing the PST have infinite length, and the PST is a statistical depiction of the construction features of the sequences.

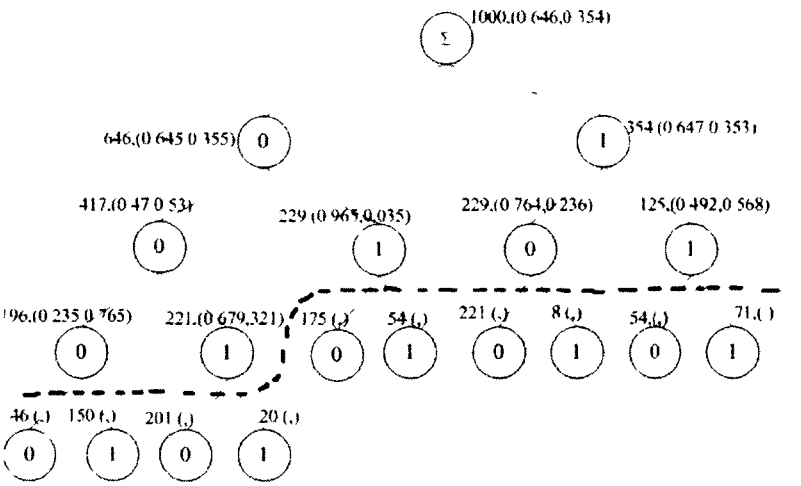

Figure 4.2: A probabilistic suffix tree

The previously mentioned significance threshold c is associated with each PST and is used to prune the tree. The pruning strategy is as follows:

1. A node whose corresponding occurrence frequency is less than c is pruned, as well as its descendants.

2. A node having no sibling is pruned, as well as its descendants.

Suffixes of rare occurrence are removed because they are statistically meaningless. Because the suffix σ associated with a node is the subsequence of the suffixes $s_i\sigma$ associated with the node's children, if σ is insignificant, $s_i\sigma$ is also insignificant, which explains pruning strategy (1). For a significant node with no sibling, the specific suffix $s_i\sigma$ dominates the set of suffixes $s\sigma(s \in \Sigma)$, and the occurrences of the rest of $s\sigma$ except $s_i\sigma$ can be ignored. Therefore, the CPD of $P(s_{s\in\Sigma}|s_i\sigma)$ can be approximated by $P(s_{s\in\Sigma}|\sigma)$, which explains pruning strategy (2). For instance, in Figure 4.2, $P(s|001) \approx P(s|01)$, the nodes '001', whose occurrence frequency is 221, are removed even though they are significant. In Figure 4.2, the dashed line prunes the bottom part of the tree if c is chosen as 55. PST pruning gives a sense to the statistical model and yields a concise tree, enabling the algorithm based on the model to work more efficiently.

The PST is an efficient tool both in construction and performance. The suffix tree can be constructed in linear time w.r.t. the length of the sequences of a cluster [81]. The main role the PST plays is to retrieve the conditional probability of $P(s_i|s_1 \cdots s_{i-1})$. We proceed as follows. We travel the tree from the root along the path root$\rightarrow s_{i-1} \rightarrow s_{i-2} \rightarrow \cdots \rightarrow s_1$, until we finish at s_1 or reach a leaf node, which means we reach a critical prefix of s_i at $s_j \cdots s_{i-1} (j > 1)$. Thus we approximate $P(s_i|s_1 \cdots s_{i-1})$ by $P(s_i|s_j \cdots s_{i-1})$; then we fetch the value of the corresponding entry of the CPD vector. For example, in Figure 4.2, the value of $P(1|001)$ can be obtained by traveling the path $\rightarrow 1 \rightarrow 0 \rightarrow 0$, stopping at the second '0' since we have reached a leaf node, so the value of $P(1|001)$ can be approximated by $P(1|01)$, i.e., $P(1|001) \approx P(1|01) = 0.236$.

4.5 Model-based K-means

Based on the new CPD model implemented via PST, we propose a novel model-based K-means algorithm for sequence clustering. Our model-based K-means avoids the need to define and calculate a pairwise measure of similarity between sequences. Instead, we use the similarity measure defined in Section 4.3, between an individual sequence and a cluster of sequences represented by a CPD model. Our algorithm differs from CLUSEQ, which builds a mode for each sequence. However, it lacks statistical significance to build a CPD model from a short sequence, like the behavior sequences from our application of personal bankruptcy prediction. Our algorithm is suitable for both long and short sequences, as our CPD model is always built from a group of sequences. The CPD model bears an analogy to the centroid in the vector-based K-means. The main idea of the algorithm is to define K CPD models to represent K sequence clusters, iteratively assign each sequence to the closest model and update the models until the membership no longer changes, which means the K CPD models become stable.

The model-based K-means for categorical sequences is described in Figure 4.3.

In the first step of algorithm described in Figure 4.3, the template PST provides a tree profile for each of the K PSTs and will be repeatedly used in step (4). As the

Algorithm: Model-based K-means for categorical sequences

1. Construct a template model (PST) over all the sequences.
2. Build K PSTs from the template PST, randomly initialize the conditional probability vectors of all the K PSTs.
3. For each sequence, calculate the similarity with each PST and assign it to the closest PST.
4. Update the conditional probability vectors of each of the K PSTs by the sequences belonging to each corresponding cluster.
   (a) Build K PSTs from the template PST.
   (b) Add each sequence to the corresponding PST.
   (c) Prune the insignificant suffixes within the PSTs.
   (d) Fill up the conditional probability vectors.
5. Repeat steps (3) and (4) until the membership no longer changes.

Figure 4.3: Model-based K-means for categorical sequences

suffixes of the K models are unknown, the structure, i.e., the depth of each leaf node, of the K PSTs is undecided. The template serves as a general frame for building K PSTs in Step (2). The template PST has no value in the conditional probability vectors. $P(s)(s \in \Sigma)$, the probability of presence of symbol s out of the total set of sequences for clustering, used to calculate the similarity by Formula (4.2), are calculated in this step. The template PST is also pruned at this step to remove suffixes that are insignificant from the global standpoint. We use the size of data set, namely the number of sequences for clustering, as the threshold c.

Step (2) is analogous to the initialization step (randomly initialize the K centroids) of the vector-based K-means. The K PSTs are built from the template PST, having the same tree structure at the beginning but different conditional probability distributions. The random values of each conditional probability vector should satisfy the requirement that the sum of the values is equal to 1.

In the third step of the algorithm, for each sequence, the similarity to each of the K models is calculated, using the similarity measure of Formula (4.2), and the sequence is assigned to the closest model. The sequences belonging to a cluster are used to update the corresponding PST of the cluster, which is implemented in Step (4).

In the fourth step of the algorithm, there are four sub-steps. These are explained as follows:

- Rebuilding the PSTs in each iteration aims to eliminate the impact in the next iteration of being over-pruned in the preceding one. Before calculating the similarity in each iteration, the PSTs are pruned to eliminate insignificant suffixes. As the sequence membership changes in the next iteration, so the structure of each PST may change, and the suffix tree from the preceding iteration may be over-pruned for the next iteration. Rebuilding the PSTs from the template PST guarantees that an earlier premature pruning will not affect the similarity calculation in the following iterations.

- Adding a sequence to a PST is done as follows: increase the occurrence frequency values of all the suffixes of the sequence prefixes by one, increase the entry of the conditional probability vector by one if the suffix is followed by the entry's corresponding symbol.[2] This procedure is run from the tail of the sequence to the head. The computational complexity of adding a sequence to a PST is linearly proportional to the length of the sequence [60, 81].

- The tree is pruned to eliminate insignificant suffixes within the cluster. Similar to step (1) of the algorithm for construction of a template tree, the significance threshold for each PST is chosen by the size of corresponding cluster.

- The trees are updated by recalculating the CPD, right after all the sequences are added to their assigned PSTs. Before updating the CPD vectors, the vectors

[2]The conditional probability vector is used to count the occurrences of the symbols following the suffix in the step of adding a sequence to a PST. And the vector is updated as a probability distribution in the fourth sub-step.

serve to store the number of presences of the corresponding symbols following the related suffix, so the vector (v_0, v_1) is updated by $\left(\frac{v_0}{v_0+v_1}, \frac{v_1}{v_0+v_1}\right)$.

4.6 Experimental results

In this section, we study the performance of the model-based K-means on both long and short sequence sets. The long sequence data set is from the SWISS-PROT protein sequence data bank, and two families, i.e., globin and immunoglobulin, are chosen to compose our experiment data set. The lengths of the protein sequences vary from 32 to 1709. The sizes of the two families are 401 and 313 respectively. The short sequence data set is from our project on personal bankruptcy prediction, and each sequence represents a behavior series of a client for consecutive 12 months, thus the length of each sequence is 12 or less (if an account has open for less than one year). The behavior sequences are composed from two symbols, i.e., '0' and '1', which means 'good behavior' and 'bad behavior', for details, see Chapter 5. The short sequence set consists of 1000 sequences representing bankrupt clients and 1000 representing non-bankrupt clients.

We compare the performance of the model-based K-means with that of CLUSEQ [93]. It is reported that CLUSEQ outperforms the sequence clustering methods based on *edit distance*, *edit distance* with block operations, q-gram, and Hidden Markov Model, so the comparison with CLUSEQ extend our comparison to these methods.

In our performance evaluation, we use precision and recall ratios of each natural class to measure the extent to which the algorithm discovers the sequence features of the class that can be used to distinguish this class with other classes (precision), and extent to which the algorithm can discover all these features (recall). The precision ratio of a class is $\frac{|F \cap F'|}{|F'|}$, and the recall ratio is $\frac{|F \cap F'|}{|F|}$, where F is the number of sequences belonging to this class and F' is the number of sequences assigned to this class. As CLUSEQ generates overlapping clustering results, we transform its results to hard partition by assigning each object to the closet cluster, which has largest similarity with the sequence. For protein sequences, a cluster is labeled as the class whose members are more than other classes in the cluster. For behavior sequences,

due to the huge imbalance of bankrupt clients and non-bankrupt clients, and our primary aim is to discover the discriminative bankruptcy feature on this data set (see Chapter 5), therefore, we are only interested in the bankrupt clusters where the sequences from bad accounts have dominant proportion over good accounts(e.g., greater than 75%).

The performance of the two algorithms on protein data is evaluated and compared by the precision and recall ratios of the two families, i.e., globin and immunoglobulin. We run both the algorithms 10 times due to the random sampling technique used to generate new cluster in CLUSEQ and random initialization in the model-based K-means. The parameters of CLUSEQ is set to K=2, significance threshold c=30 as suggested in the paper. The parameter K of the model-based K-means is set to 2. Figure 4.4 and 4.5 shows their performance measured by precision and recall respectively. In these figures, the vertical line covers the range from minimum to maximum value, and the horizontal line represents the average value over the 10 run. Specifically, their performance in average values is given in Table 4.1 and 4.2.

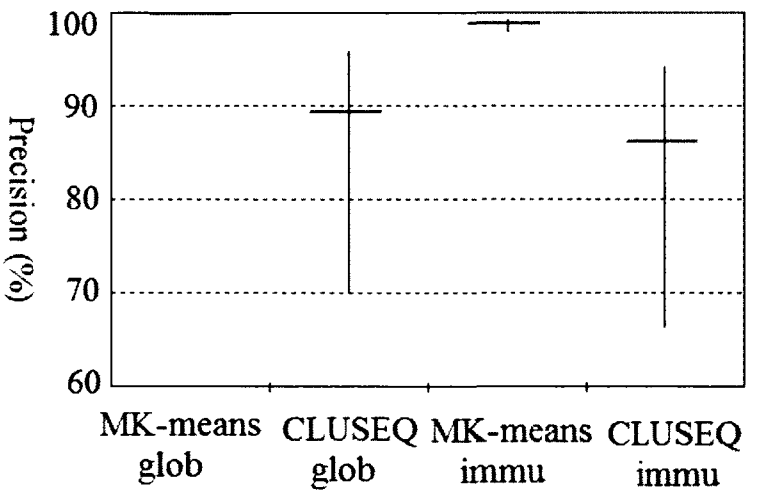

Figure 4.4: Mean precision and variation on protein data

From these figures and tables we can see that our model-based K-means outperforms CLUSEQ by a wide margin on the long protein sequences, in both clustering accuracy and stability. The results demonstrate the superior clustering performance

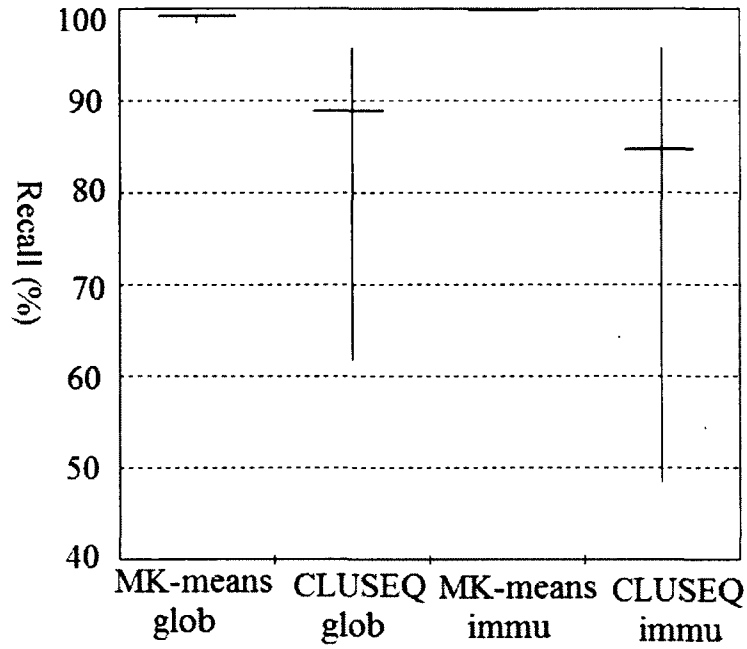

Figure 4.5: Mean recall and variation on protein data

	globin	immunoglobulin
Precision (%)	99.7	98.7
Recall (%)	99	99.7

Table 4.1: Results of the model-based K-means on protein sequences

of our algorithm, yielding clusters closely match the natural classes, with little variance in spite of its random initialization. The results of CLUSEQ show great variance due to that it uses random sampling technique to generate candidate sequences, from which, the new singleton clusters are generated. Whether the desirable new clusters can be generated in each iteration depends on whether the samples include the right choice of candidate sequences; however the random sampling cannot guarantee the yield of right choice, which leads to the unstable clustering results of CLUSEQ.

The performance evaluation on behavior sequences is based on the precision and recall ratios of the bankruptcy accounts. We run both the algorithms 10 times, as we did for protein data. We set K=5, significance threshold c=10 (as the lengths of the sequences are no more than 12) for CLUSEQ, and K=5 for the model-based

	globin	immunoglobulin
Precision (%)	89.3	86.1
Recall (%)	88.7	84.6

Table 4.2: Results of CLUSEQ on protein sequences

K-means. The precision and recall ratios of identified bad accounts by the model-based K-means are presented in Table 4.3. For CLUSEQ, the 10 runs produce the same clustering results, where the 2000 sequences are grouped into 2 clusters, and the results are given in Table 4.4.

	Precision (%)	Recall (%)
min	85.4	31.2
max	87.9	32.1
average	86.6	31.5

Table 4.3: Results of the model-based K-means on behavior sequences

Cluster	Bankrupt	Non-bankrupt
1	707	555
2	293	445

Table 4.4: Results of CLUSEQ on behavior sequences

From Table 4.3 and 4.4 we can see that our model-based K-means has great advantage over CLUSEQ on clustering short sequences. Table 4.3 demonstrates that our algorithm can discover about 30% of bad accounts with high precision, showing little variance over the 10 runs. From these slight different bankrupt clusters of the 10 runs, same sequence patterns, which are referred to as bankruptcy features, can be extracted. Chapter 5 will give more details about how our model-based K-means is used to discover prediction capable bankruptcy feature. Whereas CLUSEQ fails to identify any bankrupt cluster where the bad accounts are well separated with good ones. In CLUSEQ, the original 5 CPD models (clusters) finally consolidate to 2 models (clusters), this is because that it lacks statistical significance to construct CPD model on single short sequence, thus these models have little difference and absorb almost the same sequences to be their members, leading to the cluster consolidation.

4.7 Chapter summary

In this chapter, we investigated the challenging problem of clustering categorical sequences. We presented a statistical model, i.e., conditional probability distribution (CPD) of the next symbol given a preceding subsequence, based on which, a new measure of similarity between an individual categorical sequence and a set of categorical sequences is proposed, circumventing the obstacle of defining the naturally meaningful pairwise similarity. Based on our new similarity measure, a novel efficient and effective model-based K-means algorithm is designed for clustering sequences. The performance of our proposed algorithm was evaluated from the experiment on both long (protein sequences) and short (behavior sequences) sequence sets, the results demonstrate that our algorithm has promising utility for sequence analysis, suitable for sequences with varying length, capable of yielding superior clustering results in terms of accuracy and stability.

Chapter 5

Personal Bankruptcy Prediction

In this chapter, a personal bankruptcy prediction system running on credit card data is proposed. In our system, the bankruptcy features, which are discovered by using the clustering techniques described in Chapter 2 and 4, are employed as main predictors. The mined features are represented in low-dimensional vector space. From the new feature space, which can be extended with some existing prediction-capable features (e.g., credit score), a support vector machine (SVM) classifier is built to combine these mined and already existing features. Our system is readily comprehensible and demonstrates promising prediction performance.

5.1 Introduction

Personal bankruptcy prediction involves discovering bankruptcy features that can distinguish bad accounts from good ones. This can also be treated as a binary classification problem with two class labels, 'bad account' and 'good account'. A big challenge in building such a classification model for credit card data is that the two classes are highly unbalanced, with typically no more than 5 in 1000 credit cardholders going bankrupt. Another challenge is that the credit card data are highly multi-dimensional in that they contain numerous monthly aggregation records, which consist of various data types, including numeric attributes, discrete attributes, date attributes, and even the attributes of sequence and time series data; as well as transaction records,

which are time series data. For example, in our project on a major Canadian bank, there are more than 400 attributes in the monthly aggregation database and more than 50 attributes in the transaction database.

In our investigation, we aim to design a prediction system running on a credit card database, which is extensible, i.e., able to integrate existing prediction-capable features, either from data mining or domain expertise (e.g., credit scores); it is also readily comprehensible and can be used in industrial applications. The original purpose of our investigation was to complement existing prediction models, especially the credit scoring models, by identifying the bad accounts they tended to miss.

We explore the extent to which the use of data mining techniques described in Chapter 2 and 4, especially the sequence clustering technique in Chapter 4, can help to solve the personal bankruptcy problem. We summarize the original complex credit card data into two types: one is categorical data, which take the forms of data table with categorical variables or transaction tuples; the other is transformed behavior sequences. The applications of our data mining techniques in personal bankruptcy prediction are described in Section 5.3 and 5.4.

The final prediction system is built on a support vector machine (SVM). We use the SVM as the final classifier because it is widely accepted that the SVM is one of the most robust and accurate classification methods of all the well-known algorithms, and can be easily extended to output continuous values to rank the instances on the likelihood of belonging to the positive class [20, 85]. The bankruptcy features obtained from our data mining approach are represented in low-dimensional vector space, which is in the form of Boolean table, can be extended with some existing prediction-capable features (e.g., credit bureau credit scores). Finally, the SVM classifier is built on the comprehensive feature space. The structure of our prediction system is illustrated in Figure 5.1. In our system, the SVM is used to combine the bankruptcy features, and to output a confidence score for each client, indicating the likelihood of being a bad account. Note that each bankruptcy feature alone has great prediction capability and easy to understand, thus the use of SVM does not decrease the comprehensibility of our prediction system; this distinguishes our system from most existing systems or models, whose classifiers are built directly on aggregated vector space.

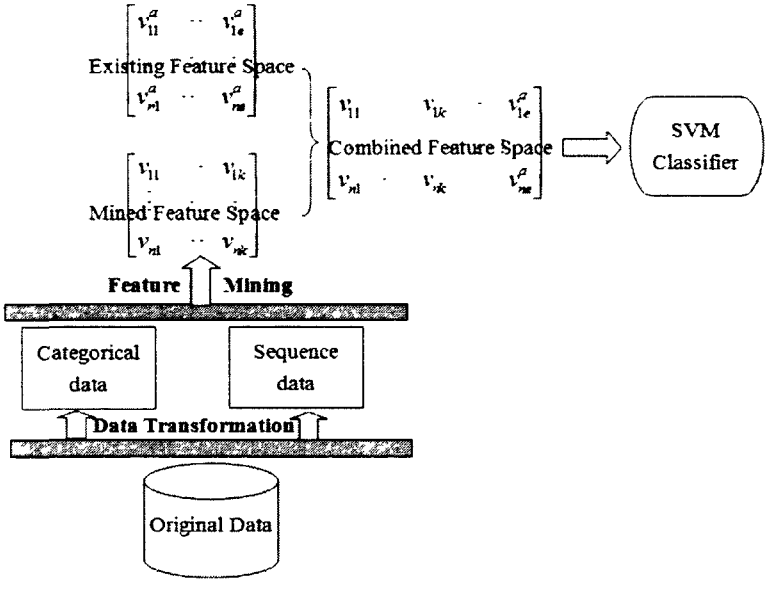

Figure 5.1: Framework of our prediction system

5.2 Related work

Personal bankruptcy is a phenomenon that is difficult to understand. There is little supporting theory related to this phenomenon. Compared to business clients, personal clients have much larger volumes of data, and it is difficult to resort to experienced and informed risk evaluation for predicting personal bankruptcy. The most commonly used methods in personal bankruptcy prediction are credit scoring models [42, 47, 52, 61, 79], especially the generic scoring models developed by credit bureaus (Equifax, TransUnion and Fair Isaac). These models are based on empirical knowledge, as they are developed by analyzing statistics and picking out characteristics that are believed to relate to creditworthiness.[1] Credit scoring systems, especially the generic ones, are run on multiple data sources from many creditors. In making a final decision, a customized system developed by the individual creditor is usually combined with generic scores purchased from a credit bureau.

Because of the high-dimensional character and complexity of the data, there is no existing data mining model that can handle all the data at one stroke. While it

[1]Credit Scoring. Available: `http://www.epic.org/privacy/creditscoring/`

is well known that some data mining models, such as decision trees, neural networks and support vector machines, are applied to this domain, there is very little published literature concerning personal bankruptcy prediction in the data mining field. This is mainly due to two things. The first is commercial-in-confidence: corporations do not like to reveal their techniques to others. Second, large and interesting sources of data are not made available to the academic community. Even several of the papers that have been published [28, 42, 67] do not present a practical prediction system, but just give some classification models where the prediction is simply treated as classification, without considering the prediction period, which is a critical issue for practical application. These existing models use a fixed-dimension vector consisting of "those attributes that were found to be most correlated to bankrupt behavior" [28] to represent a client, and the prediction models are trained in the vector space.

Comprehensibility is another issue that creates a gap between the academic and industry communities in the field of personal bankruptcy prediction. Both accuracy and comprehensibility are required for data mining techniques [80]. The patterns discovered by the data mining models have little value in practical application if they have low comprehensibility. The creditor hesitates to use a complex prediction model with low comprehensibility, even if its prediction accuracy is high. In summary, some data mining models concerning personal bankruptcy prediction have several drawbacks:

- The attributes fed into the models need to be pre-selected, which is difficult even with knowledge and experience of the domain.
- The format of data input into the models is vectorial, which means that original sequence data need to be aggregated into one value for each attribute, and this kind of aggregation leads to significant loss of useful information, such as sequence and sequential patterns.
- The models can be difficult to interpret to creditors, especially if they are not experts in the data mining field.

Thus, employing these models directly as classifiers is difficult in industry applications. Furthermore, although these models have demonstrated some capacity to outperform basic classifiers such as decision trees, neural networks, etc, their predictive performance is not very satisfactory. For instance, in the hybrid model of Donato [28], 63.7% of bankrupt (bad) accounts are identified, with 17.3% of non-bankrupt (good) accounts mistaken for bad accounts; while in He's MCNP model [42], 82.8% of bad accounts are identified, but the percentage of good accounts that are misidentified is as high as 49.0%. These two systems lack appeal, due either to a low true positive ratio or a high false positive ratio.

5.3 Feature mining from categorical data

The extracted or transformed categorical data in our project took two forms: data table with categorical variables and transaction tuples. Categorical data sets were extracted from the original monthly aggregation database, and transaction tuples were obtained by aggregating daily transaction record database to summarize which kinds of merchandise or service each client purchased using the credit card. The summarization for transaction is based on pre-defined taxonomy, for example, purchasing airline tickets from different airline company can be summarized to 'air transportation' category or even 'transportation' category, depending how fine the taxonomy is.

From the original monthly aggregation database, we obtained two kinds of categorical variables. One is stationary categorical variables, whose values do not change or change little within past historic period used for model learning, such as age, gender, province, employer etc. The other is transformed from monthly numeric data using discretization method. For example, the values of "cash advance" variable can be divided into three categories: 'no cash advance', 'less than 1000$', and 'equal or more than 1000$'. The symbols used to denote the discretized categories are natural numbers, i.e., {0, 1, 2, ...}. Our investigation found that these stationary categorical variables did not provide discriminative information that can distinguish bad accounts from good ones. We thus built the categorical table just from the discretized

numeric variable, where each client was represented by several categorical vectors corresponding to the historic period used for prediction. Figure 5.2 gives an example of the representation of a client in the categorical table. Some categories from different variables may show strong discrimination power when they join together, such as the category 'equal or more than 1000$' from 'cash advance' variable occurs together with the category 'equal or more than 500$' from 'balance increase' variable in a month. DHCC discovered one such prediction capable feature from our preliminary transformed categorical table, which was used as one of the final predictors. Its discriminative power is shown in Table 5.3 in Section 5.6.

ID	MONTH	Bankrupt_Label	V1	V2	V3	V4
99753	200603	1	1	0	1	0
99753	200602	1	1	2	0	0
99753	200601	1	1	0	0	0
99753	200512	1	1	0	0	0
99753	200511	1	1	2	0	0
99753	200510	1	0	0	2	0
99753	200509	1	1	0	1	0
99753	200508	1	1	0	1	0
99753	200507	1	0	0	1	0
99753	200506	1	0	1	0	0
99753	200505	1	0	0	0	0
99753	200504	1	1	0	0	0

Figure 5.2: Categorical table built from credit card data

We also explored if the pattern discovered from transactional data can be used to distinguish bad accounts from good ones. The merchandise and service that purchased by the credit cardholders were summarized by 14 categories, denoted by {A, B, C, ..., N}. Each client is represented by a transaction, i.e., a subset of the 14 items, denoting the past consumption. The clustering results generated from the transactional data are presented in Table 5.1, from which we can see that cluster '111' represents a bankruptcy cluster as the majority of the transactions are from bankrupt accounts. The bankruptcy feature we extracted from cluster '111' is that using the credit card mainly purchase category 'M', with very few other items, where the category 'M' is insurance. However, this feature is not selected in our final prediction system due to its relatively lower discriminative power and low coverage of bad accounts (small recall value) compared with other mined features.

Cluster	#Bankrupt	#Non-bankrupt
000	129	201
001	220	210
010	173	377
011	306	394
100	477	328
101	189	185
110	230	229
111	**276**	**76**

Table 5.1: Clustering results of DHCC on transactional data

5.4 Feature mining from sequence data

5.4.1 Motivation

To the best of our knowledge, in the personal bankruptcy prediction domain, neither the scoring models nor the existing data mining methods adequately take sequence information in credit card data into account. Sequence data are very common in data sources such as credit card data, which takes the form of ordered monthly aggregation or transaction records with a time stamp. The evolutionary change of value in this kind of sequential data or time series data contains strong predictive patterns: for example, a period of consecutive increases in balance is an indicator of a risky account. At the same time, original temporal data directly extracted from individual numeric variable yield short time series, since in financial institutes, the past historic data used as a predictor normally spans one year [28, 42, 67]. Thus the number of data points in the time series is about 12. This causes difficulties for conventional time series analysis methods which do not work well on such short time series.

Instead of aggregating the original sequential and time series data into vectorical data, we use sequence mining technique to discover the bankruptcy features. We exploit the clustering technique described in Chapter 4 to discover useful sequence patterns which can be used to distinguish bad accounts from good ones. To make our prediction system comprehensible, we do not build the classifier, which is commonly used to make predictions, directly on the sequences. Instead, we exploit sequence

clustering to discover the sequence patterns, which are easy to understand, and use them as predictors in the final prediction system. Additionally, building the sequence classifier requires definition of the similarity between the sequences, which is difficult in our application, as our sequences are short and contain noise; furthermore, the sequence patterns vary for different variables. On the other hand, clustering is used to find useful variables on which predictive sequence patterns can be mined. Therefore, clustering is used as an exploratory tool to discover comprehensible predictors for our final prediction system.

5.4.2 Sequence representation of client behavior

Each variable, which can be an original attribute or a function of an original attribute or attributes, or an aggregation of original data, results in a set of sequences (for different clients). In general, for categorical variables, the sequences can be constructed directly by extracting categorical values from the original database. For a numeric variable, discretization is employed to transform the variable to a categorical one before constructing sequences. The process of these transformations can be considered as coding human knowledge about the information provided by each variable. In our application, each original variable is transformed either to a binary categorical variable (meaning 'good behavior' and 'bad behavior') or a multi-valued ordinal attribute (meaning 'good behavior' and 'graded bad behaviors'). The symbols used to construct the sequence are natural numbers, i.e., {0, 1, 2, ...}. The symbol '0' represents good behavior (e.g., no delinquency), while non-zero symbols represent bad behavior (e.g., delinquency); the 'bigger' the symbol, the worse the behavior. For instance, coding '0' if no cash advance in a month and '1' otherwise; coding '0' if there is no delinquency in a month, '1' for slight delinquency [2], and '2' if a serious delinquency [3].

Consequently, we obtain two types of sequences, i.e., binary sequences and ordinal

[2]Slight delinquency means the client failed to pay the minimum required amount due in a month by the due date but paid by the next billing date; this is defined in the original data.

[3]Serious delinquency means the client failed to pay the minimum required amount due in a month by the next billing date; this is a new variable created in our system.

sequences. If the variable has two categorical values, the resulting sequences are binary; if the variable has more than two values, they are ordinal. The specific characteristics of these two types of sequences are as follows:

- The symbol set composing the sequences is asymmetric; i.e., only the presence of non-zero symbols is regarded as an important bankruptcy indicator, but this does not mean the presence of '0's can be ignored. In fact, these '0's characterize the environments surrounding non-zero symbols; the same symbol(s) in different environments represent different behaviors. Taking '0000011000' and '00101101' as examples, the two clients clearly have different levels of delinquency since the segment '11' is located between many 0's in the first sequence and only two 0's in the second.

- The symbols composing the ordinal sequences are comparable in the following way: '2' is worse than '1' which is worse than '0'. On the other hand, the symbol '1' is more similar to '0' than to '2' in some cases, but the reverse may be true in other cases.

Generally speaking, the scheme for converting original data into sequence data is straightforward. The natural-number-denoted sequence symbol represents valuated monthly behavior, which simplifies the original data while preserving the sequential or temporal information.

5.4.3 Modification and extension of the CPD model

We make some modification and extension of the CPD model to adapt it to the special character of our sequences. The weighted similarity measure is designed for binary sequences and applied to ordinal sequences as well because ordinal sequences are decomposed to binary sequences in our clustering algorithm (details are given in this section). The new measure is given in (5.1), which is derived from the one defined

in (4.2).

$$
\begin{aligned}
sim_{\Delta}(S) &= \prod_{i=1}^{l} \exp\left(w_i \left(P_{\Delta}(s_i|s_1 \cdots s_{i-1}) - P(s_i)\right)\right) \\
&= \frac{\exp\left(\sum_{i=1}^{l} w_i P_{\Delta}(s_i|s_1 \cdots s_{i-1})\right)}{\exp\left(\sum_{i=1}^{l} w_i P(s_i)\right)}
\end{aligned}
\tag{5.1}
$$

where

$$
w_i = \begin{cases} w_0, & \text{if } s_i\text{='0'}; \\ w_1, & \text{if } s_i\text{='1'}. \end{cases}
$$

Here w_0 and w_1 are the weight values for '0' and '1' respectively. In our application, we choose w_0=1.0, and w_1=2.0.

We emphasize the structural character of non-zero symbols by using a weighted similarity measure. Because the symbol set composing the sequences is asymmetric, the probability of generating zero symbols and non-zero symbols should not play equally important roles in the similarity measure. For example, the sequence '000000011111' may be assigned to a group of sequences with sparse occurrence of non-zero symbols, like '0000000100'. But as the local structural character of non-zero symbols plays an important role in identifying bad accounts with a high risk of going bankrupt, we are more likely to assign the sequence '000000011111' to the cluster of sequences with frequent and consecutive occurrence of non-zero symbols like '11111011011'. So in our application, w_1 should always be greater than w_0.

In order to extend the similarity measure defined in (5.1) and the model-based K-means algorithm in Chapter 4 to ordinal sequences, we decompose the ordinal sequence to a binary sequence set consisting of m binary sequences. The hierarchical decomposition is associated with the ordinal relationship of the symbols. The binary sequence of order i takes the value of '1' at a given position if one of the symbols from α_i to α_m is present at that position, here α_i is the symbol of natural number i, and '0' otherwise. Occurrences of '1' in the higher-order sequence are repeated in lower-order sequences at the same position. An example is given in Figure 5.3. For the sequence whose 'biggest' symbol is smaller than α_m, the binary sequence of high

order takes the value of '0' at all positions. There are m CPD models, as well as m PSTs, for a cluster of sequences, each relating to one order. Each PST is constructed by the decomposed binary sequences of the related order.

Original sequence:	**000000102223**
First order:	**000000101111**
Second order:	**000000001111**
Third order:	**000000000001**

Figure 5.3: Decomposition of an ordinal sequence

We define the measure of similarity between an ordinal sequence and a cluster of ordinal sequences based on a weighted combination of the similarity between each binary sequence and the CPD model of the same order. We use ss_k to represent a (user-defined) similarity between two neighbor symbols, i.e. ss_1 is the similarity between '0' and '1', ss_2 between '1' and '2', etc. The symbol '1' in the sequence of higher order represents worse behavior, which means the dissimilarity between the non-zero symbol and the symbol '0' plays an important role in identifying bad accounts. Using $d_j = \sum_{i=1}^{j} \frac{1}{ss_j}$ to quantify the dissimilarity between symbol α_j and '0', the similarity measure for ordinal sequences is defined as:

$$sim'_{\Delta}(S) = \frac{1}{\sum_{j=1}^{m} d_j} \sum_{i=1}^{m} d_i sim_{\Delta_i}(S_i) \tag{5.2}$$

where S_i is the binary sequence of S discomposed on order i, Δ_i is the binary sequences of all the sequences of Δ discomposed on order i, and $sim_{\Delta_i}(S_i)$ is the similarity between S_i and Δ_i as defined in (5.1). The higher the order i is, the more heavily the similarity $sim_{\Delta_i}(S_i)$ is weighted.

The model-based K-means clustering algorithm for ordinal sequences is almost the same as the algorithm described in Figure 4.3, except that the template model has m PSTs and each model of a cluster has m PSTs. In step (3), the similarity between an individual sequence and a cluster of sequences is calculated by Formula (5.2). In step (4), the CPD of a cluster is updated, recalculating the CPD of each

PST based on the binary sequences of the corresponding order.

5.4.4 Sequence pattern extraction

Sequence patterns are extracted based on the clustering results of each variable. This process is done manually to make the discovered patterns easy to understand for creditor. This exploratory analysis process involves, aggregating the patterns hidden in several different clusters if it can, and merging some adjacent categorical values if possible. This process proceeds as follows: First, we explore the model-based K-means clustering (trying different values for K) to find bankruptcy clusters where the majority of sequences are from bad accounts, and then we analyze and extract the common structural features of the sequences, referred to as bankruptcy features, in bankruptcy clusters.[4] Table 5.2 shows an example of the clustering results for 'cash advance' variable with K=6; clusters 2 and 6 are bankruptcy clusters. Some samples from these two clusters are depicted in Figure 5.4. By analyzing the structural features of the sequences in these two bankruptcy clusters, we can aggregate the sequence patterns in the two clusters, concluding the final sequence pattern as 'there is cash advance for more than 2 consecutive months or 2 consecutive months with at most 1 preceding and following month with no cash advance'. Then, the features' capacity to distinguish bad accounts from good ones is tested; if the accounts identified by these features are close to the ones in related bankruptcy clusters, the features are chosen as bankruptcy features.

Cluster	#Bankrupt	#Non-bankrupt
1	998	1412
2	**512**	**108**
3	160	109
4	92	182
5	108	154
6	**130**	**35**

Table 5.2: Sequence clustering results for one variable

[4]Fortunately, clustering for finding bankruptcy clusters is not sensitive to K: the same bankruptcy features can be extracted for several different choices of parameter K.

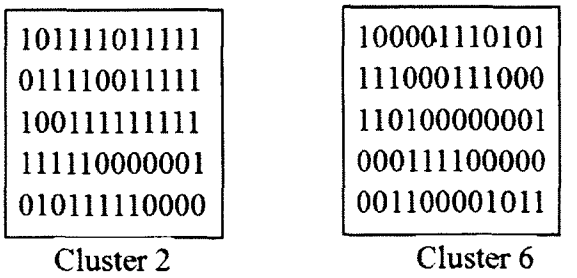

Figure 5.4: Samples drawn from two bankruptcy clusters

In addition, by analyzing the other aspect of sequence patterns, some adjacent categorical values can be merged if they represent similar behavior to some extent. Figure 5.5 shows some samples from two non-bankruptcy clusters on another variable, from which we can see that the categorical value '1' can be merged to '0' as it constructs non-bankruptcy patterns similar to those for '0'. When some adjacent values are merged, which means the attribute is re-discretized to less categorical values, the newly generated sequences on the attribute are re-clustered. By aggregating patterns and merging similar categorical values, the final patterns extracted from the clustering results are simplified. Since both the sequence coding scheme and the statistical measure of similarity are readily comprehensible, the bankruptcy features obtained are also easy to understand, as it is shown in the example of the cash-advance feature.

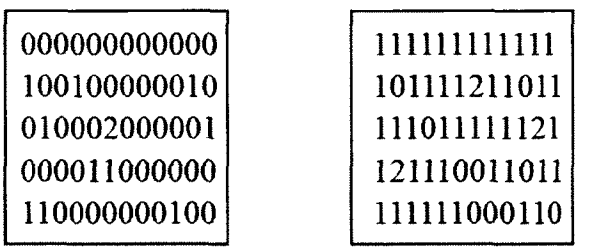

Figure 5.5: Samples drawn from two non-bankruptcy clusters

5.5 Bankruptcy feature representation

The bankruptcy features, including the features mined from our categorical data mining and sequence mining techniques and the features from other data mining models or domain expertise, are represented in low-dimensional Boolean vector space. The number of the dimensions is the number of the selected prediction-capable features. The clients are represented in the new readily comprehensible vector space. For each dimension, the vector takes value 1 if the client has the corresponding bankruptcy feature and value 0 otherwise. The Boolean table combines the knowledge mined from the original high-dimensional, complex credit data. Instead of binary classification, i.e., predicting whether each client will go bankrupt or not, we combine these mined prediction-capable features to generate a confidence score for each account. The SVM classifier is exploited for this purpose, with binary value {+1,-1} indicating class membership, +1 for a bankrupt account and -1 for a non-bankrupt account. Thus, the higher the confidence score, the more risky the account is.

Up to this point, we have transformed the original classification problem on a complex, large data base to classification on a concise, low-dimensional Boolean table. Regarding the function of the final classifier, we emphasize the aspect of combining the predictive-capable features to output a continuous score, in order to rank the accounts according to the probability of going bankrupt. It is impossible to design a classifier on the original high-dimensional sequential and time series data.

The bankruptcy patterns discovered by our data mining techniques are relatively stable. The process of data clustering and feature extraction is an independent part of our prediction system, which can run independently alongside the prediction system. In our application, with the observation point moving forward for half a year,[5] the same bankruptcy features can be obtained from our data mining technique. Therefore, the mined bankruptcy patterns just need to be tracked to see if their predictive capability is changing, and turn off the related dimension in the feature space if the corresponding feature is prediction-incapable. Additionally, the feature space is

[5]The credit card data delivered to us cover for two and half years. Predicting the future bankruptcy for one year using preceding one year's historical data requires the period for two years, so only half a year remains allowing to test the stability of the mined bankruptcy features over time.

extended if a new feature was discovered in the latest period.

5.6 Prediction results

Our prediction system is based on the combined feature space of the mined bankruptcy features and some existing prediction-capable features, such as credit bureau credit scores. To demonstrate the prediction performance of our data mining technique, in Section 5.6.1, we give the experimental results based on the mined bankruptcy features, and compare them with the performance of credit bureau credit scoring and that of a SVM built on simple aggregation of original data. We also present the prediction results of a decision tree based on both the mined bankruptcy features and the simple aggregation data. Additionally, in Section 5.6.2, we present promising prediction results from our final prediction system, which is built on the mined bankruptcy features and some already existing prediction-capable features.

The data used in our experiment is from our project for one of the major banks of Canada. There are a total of 7495 bad accounts among the MasterCard holders, all from bankrupt clients whose accounts were activated before April 2006 and who declared bankruptcy in the period from April 2006 to March 2007 (one year). We also include 10,175 good accounts in the experimental data set. The period of the data used as predictor is from April 2005 to March 2006 (one year), i.e., March 2006 is the observation point. So, our personal bankruptcy prediction system uses one year of historical credit data to predict future bankruptcy for one year. The training set contains both 2000 bad accounts and 2000 good accounts, which are randomly selected. The remaining 5495 bad accounts and 8175 good accounts are used to evaluate the prediction performance.

In our investigation, we use three percentage values to evaluate our prediction system. The first one is the hit ratio: the number of identified bad accounts over the total number of bad accounts; the second is the ratio of the total balance of identified bankrupt accounts to the total balance of all bankrupt accounts; and the third is the misidentification ratio, i.e., the ratio of the number of misidentified good accounts to the total number of good accounts. Receiver Operating Characteristics (ROC) curves

are exploited to evaluate performance. The main difference between the measures used in our work and conventional measures such as Type I and Type II error rates, as used in [67], is that the prediction performance is evaluated separately by us for the class of bad accounts and the class of good accounts. In fact, in the real-life situation the overwhelming majority of accounts are good clients (about 0.5% of clients declare bankruptcy each year in our project). Thus the type I or type II measure, calculated by combining the number of identified bad accounts and the number of misidentified good accounts, would yield biased indications when the proportions of the two classes in the experiment do not reflect the real situation. Given the huge imbalance of the classification, if $\alpha\%$ of good accounts were misidentified, this would approximately be $\alpha\%$ (slightly higher) of total accounts that are treated as risky accounts, since the number of bankrupt accounts can be neglected in such ratio calculation. This is the reason why it is important to considerer the two classes separately, with the goal of maximizing the hit ratio for bad accounts while simultaneously minimizing the misidentification ratio for good accounts.

We compare our prediction results with the performances of both the SVM built on simple aggregation of the variables and credit bureau credit scoring, and also, we give the prediction results of decision tree based on both the mined bankruptcy features and the simple aggregation data. For the SVM, the input vectors are generated from the capable variables selected by our clustering algorithms; the vector value of each variable is the average of the historical data for the preceding 12 months. Here, the goal of comparing with SVM is to show the importance of considering comprehensible features, especially the sequence patterns. On the other hand, the comparison with the generic scoring models from three major credit bureaus [6] is also very important because these are preferred and widely used in practice in the industry. In fact, these models have been expanded well beyond their original purpose of assessing credit risk to predict potential bankruptcy, profitability, fraud etc., see the literature on the

[6]Trans Union, Equifax, and Northern Credit Bureaus, Inc. are the largest credit bureaus providing credit report services in Canada. They get information from companies granting credit and public record information from courthouses throughout Canada. Their systems are complex and the details are not revealed to the public due to commercial-in-confidence. The monthly credit score is part of our original data.

Internet [7] and in [61]. We used the SVM with the Radial Basis Function (RBF) kernel from LIBSVM [8] for both our final prediction system and the SVM classifier built on simple aggregation for comparison. The decision tree is constructed by the C4.5 algorithm [68], using the entropy-based criterion to select the split. For the decision tree, the predictors are categorical in the bankruptcy feature space, continuous in the aggregation data space.

5.6.1 Feature mining

Fourteen bankruptcy features, denoted as v_1, v_2, ..., v_{14}, are identified as capable bankruptcy predictors. The first thirteen features are discovered by mining behavior sequence, and the last one is discovered from categorical data. Their names are omitted here due to commercial-in-confidence. Their prediction performance is given in Table 5.3, where the column 'bad' gives the number of bad accounts identified, the column 'good' gives the number of good accounts misidentified, and '%bad' and '%good' are the respective percentages of right prediction (True Positive) and wrong prediction (False Positive). The cardinality of the union of bad accounts hit by these 14 variables is 5112, accounting for 93.03%, and the number for good accounts misidentified is 2133, accounting for 26.09%.

Figure 5.6 shows the distribution of the identified bad accounts in Table 5.3, referring to the 5112 bad accounts hit by either of the 14 bankruptcy features, over the 12-month prediction period. From Figure 5.6 we can see that more than 95% of the bad accounts declared bankruptcy within one month after the observation point are identified, and the ratio decreases as the prediction period is extended. For cases of bankruptcy occurring in the twelfth month after the observation point, about 87% of the bad accounts are identified. As we explained in Chapter 1, the earlier bad accounts are identified, the lower the losses entailed. However, early identification represents a greater challenge, which is illustrated in Figure 5.6.

Objective comparisons were conducted based on the ROC curves obtained from our model, i.e., the SVM built on the mined bankruptcy feature space; the SVM (built

[7] Credit Scoring. Available: `http://www.epic.org/privacy/creditscoring/`

[8] LIBSVM. Available: `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`

Variable	bad	good	%bad	%good
v_1	1691	90	30.77	1.10
v_2	1810	681	32.94	8.33
v_3	1203	513	21.89	6.28
v_4	1676	655	30.50	8.01
v_5	1157	425	21.06	5.20
v_6	2839	813	51.67	9.94
v_7	1462	631	26.61	7.72
v_8	2081	495	37.87	6.05
v_9	3056	627	55.61	7.67
v_{10}	1416	698	25.77	8.54
v_{11}	3157	614	57.45	7.51
v_{12}	1137	169	20.69	2.07
v_{13}	1270	429	23.11	5.25
v_{14}	1089	352	19.82	4.31
Union	5112	2133	93.03	26.09

Table 5.3: Prediction results for capable variables

on simple aggregation of original data); and credit bureau credit scoring. The ROC curve for case identification, i.e., the hit ratio (True Positives) vs. misidentification ratio (False Positives) is displayed in Figure 5.7. From Figure 5.7 we can see that our model performs very comparably to credit bureau credit scoring. At low false positive, the credit scoring system outperforms our sequence model by identifying slightly more bad accounts; however, at higher false positive, our sequence model overtakes the credit scoring. Comparisons using two cutoffs for the false positive value, 10% and 15%, are given in Table 5.4. Of the 7495 bad accounts, 381 have account records for only 6 months or less; these accounts were opened after September 2005 but went bankrupt before March 2007; also, some bankrupt clients have only used this credit card very few times or even not at all. These two kinds of accounts are almost impossible to identify using credit card data sources alone. This situation gives some advantages to the credit scoring method because it is based on historical credit information from almost all creditors, even from public sources like court cases. Thus the credit bureaus may have plentiful predictive information on these accounts, which are difficult for us to identify.

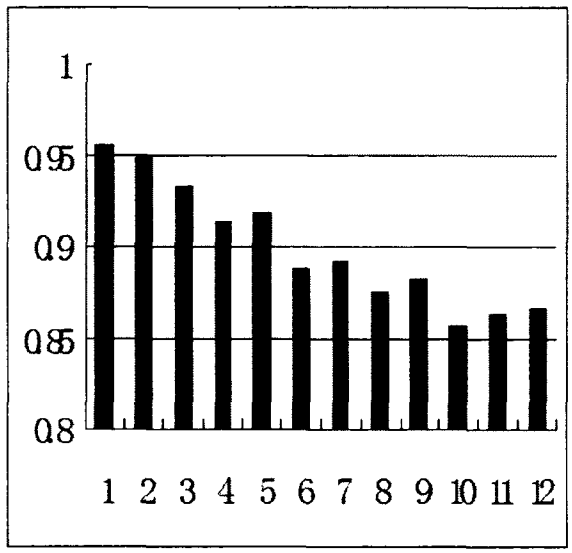

Figure 5.6: The distribution of identified bad accounts over the prediction period

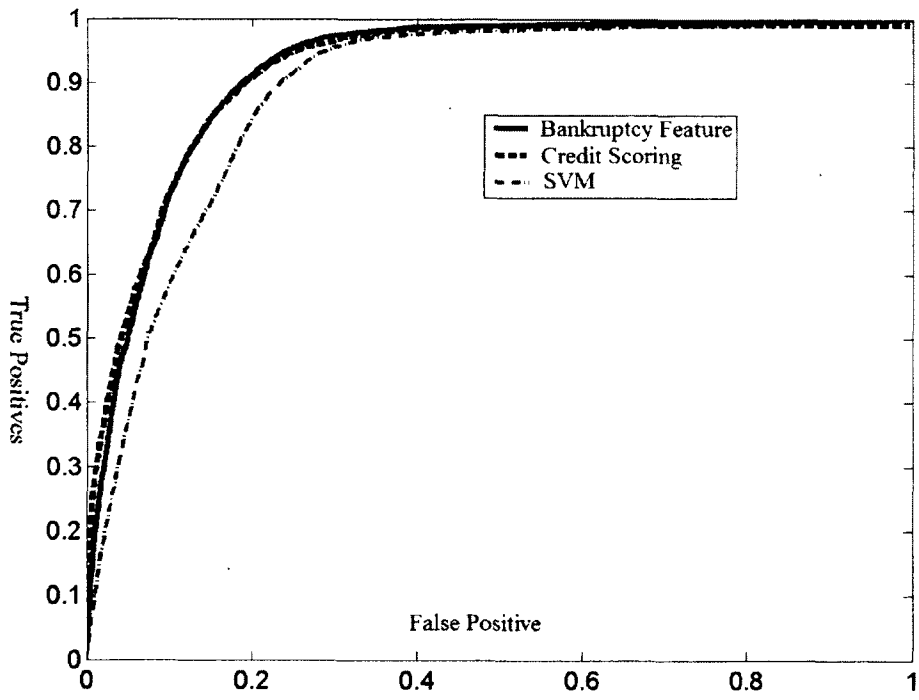

Figure 5.7: ROC curve for case identification

Cutoffs of False Positive	10%	15%
Our System	72.35%	84.46%
SVM	58.49%	70.94%
Credit Scoring	72.68%	84.59%

Table 5.4: Case identification on two cutoffs

The prediction results of the decision tree based on both bankruptcy features and simple aggregation are given in Table 5.5. The ROC curve cannot be calculated for decision because the output of the decision tree is classification, which cannot be used to rank the accounts. The decision tree can be extended to the ranking problem, provided that the ordinal scale for each instance is supplied [86]. However, the ordinal scales are unavailable in our application; therefore, for each test on either features or simple aggregation, we can only get two ratios, i.e., true positive and false positive. For comparison, in the False Positive column, the value of the decision tree is followed by the value of our system under the approximate true positive value.

	False Positive	True Positive
Feature Space	18.57% / 14.82%	83.97%
Simple Aggregation	37.7% / 26.9%	96.4%

Table 5.5: Prediction results of decision tree

Table 5.5 also demonstrates that our mined bankruptcy features are more prediction-capable than simple aggregation. The performance of a decision tree on the feature space is slightly inferior to that of our system, while our system outperforms the decision tree on simple aggregation by a wide margin.

In our evaluation, we have used the ratio of the balance of identified bankrupt accounts to the balance of all bankrupt accounts. This is an important measure for evaluating the performance of a prediction system, since the primary goal of personal bankruptcy prediction is to lower the losses resulting from personal bankruptcy. Two time points are important to take into account: one is the observation point, which is March 2006 in our experiment, and the other is the time when the client declared bankruptcy, which varies with different bankrupt clients. The ROC curve of fraction of balance of identified bankrupt accounts on March 2006 vs. the fraction of the

number of falsely identified good accounts (False Positives) is given in Figure 5.8, and the ROC curve of fraction of balance of identified bankrupt accounts at the time the clients declared bankruptcy vs. the fraction of the number of falsely identified good accounts (False Positives) is given in Figure 5.9.

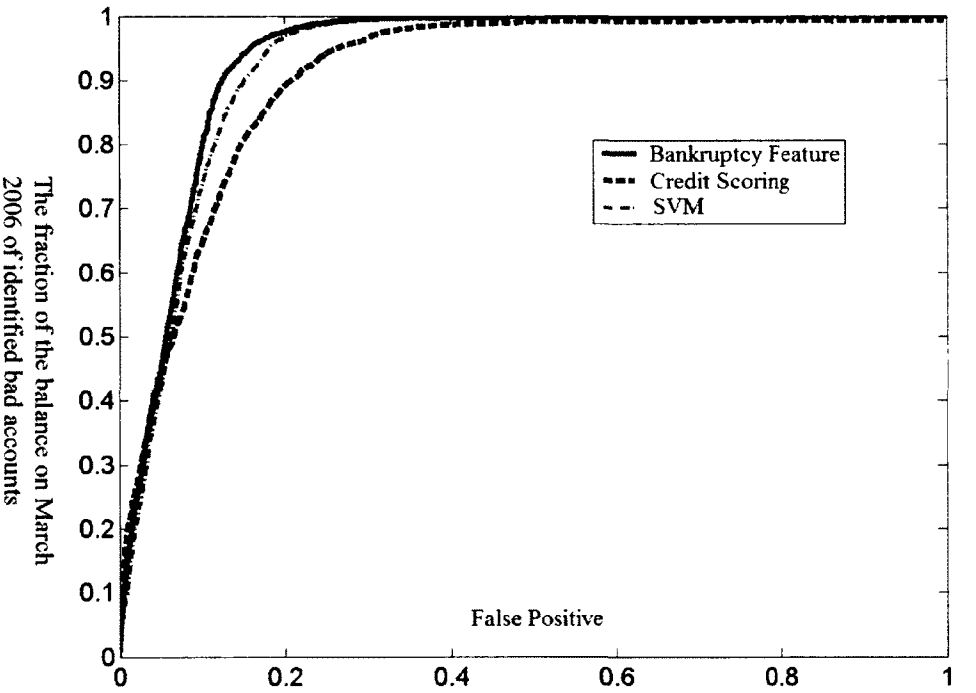

Figure 5.8: ROC curve of March 2006 balance of identified bad accounts

Both Figure 5.8 and Figure 5.9 show that our system outperforms credit bureau credit scoring and the SVM, notably in terms of loss prediction. For example, our system can predict 94.07% of the total balance of the bankrupt clients at March 2006 and 90.90% of the total balance of the bankrupt clients when they declared bankruptcy when the misidentification ratio is 15%, compared with 89.37% and 84.9% for the SVM, and 80.54% and 78.02% for credit bureau credit scoring. These results are shown in Table 5.6. The loss prediction comparison for the false positive cut-off of 10% is given in Table 5.7. These results show that our system can identify more critical-bankrupt accounts, i.e., those bankrupt accounts owing large amounts

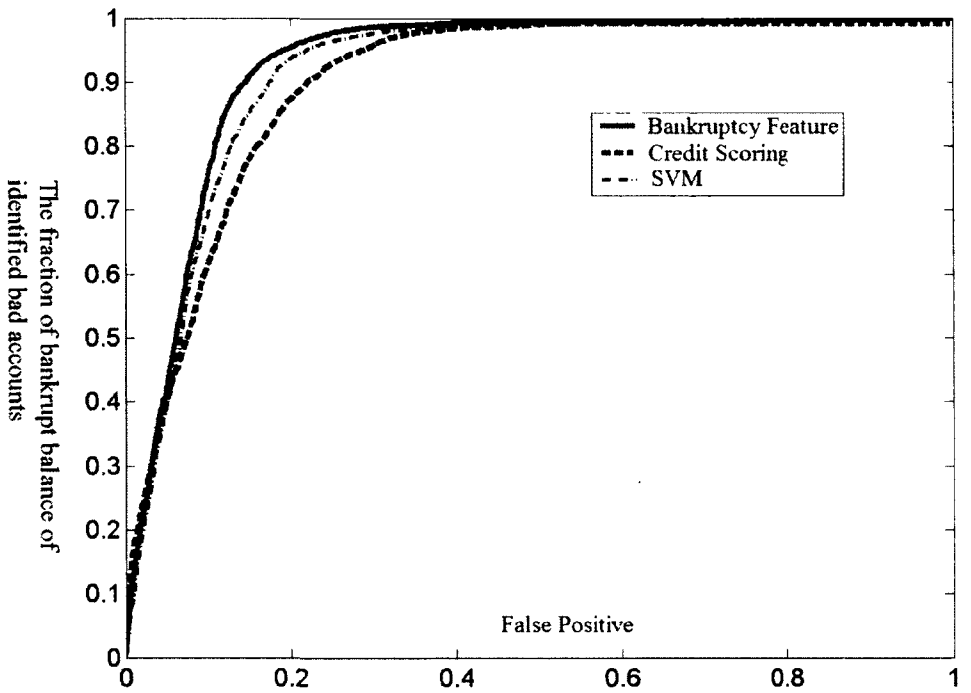

Figure 5.9: ROC curve of balance of identified bad accounts when bankruptcy declared

of money. This is reasonable because clients with large credit scores are granted high credit limits, so that when these clients go bankrupt, they owe large sums to their creditors. These clients cannot be identified by the credit scoring system; however, most of them can be identified by our system as they have the bankruptcy features we mined. Therefore, our system can be used as an important supplement to credit scoring systems.

	Bal_Observation Point	Bal_bankrupt
Our System	94.07%	90.90%
SVM	89.37%	84.9%
Credit scoring	80.54%	78.02%

Table 5.6: Loss prediction for 15% cutoff

	Bal_Observation Point	Bal_bankrupt
Our System	80.28%	76.17%
SVM	73.77%	69.57%
Credit Scoring	64.65%	61.55%

Table 5.7: Loss prediction for 10% cutoff

5.6.2 Final prediction system

The aim of our research is by no means to design a prediction system to replace existing systems, but rather to extend current approaches to improve prediction performance. The feature space for building the final SVM classifier is extensible. With more prediction-capable features, the final prediction performance can be improved. Our final prediction system combines the bankruptcy features obtained by our data mining techniques and two existing features that are capable of distinguishing bad accounts from good accounts. These two existing features are referred to as non-bankruptcy features because the presence of these features indicates that an account has little risk of going bankrupt. The two non-bankruptcy features are below.

The first is the credit bureau credit score, a preferred measure widely used in industry. A credit score is a number, generally between 300 and 850, assigned to a

client to rate creditworthiness. The scores are assigned based on multiple sources, such as the mortgage industry, the banking industry, etc. Generally, a score above 750 is considered to be excellent. [9] Therefore, we use 750 as the cutoff score: a client whose credit score at the observation point (March 2006 in our experiment) is above 750 possesses this extended feature, that is, the vector corresponding to this client takes value 1 on this extended dimension; otherwise, it takes value 0.

The other feature used to extend our prediction system is good payment history. A client's payment history is believed to be a good indicator of future insolvency. A client who has never been delinquent is considered to represent little risk. A client who never commits delinquency possesses this extended feature, that is, the vector corresponding to this client takes value 1 on this extended dimension; otherwise, it takes value 0.

The final prediction performance based on the extended feature space is shown in Table 5.8, which is based on the two false positive cutoffs, i.e., 10% and 15%. Comparing with Tables 5.4, 5.6 and 5.7, we see that notable improvement has been gained by extending our mined feature space with some existing features. Furthermore, when the credit bureau credit score is combined with our final prediction system, our system outperforms credit scoring by a larger margin.

Cutoffs of False Positive	10%	15%
Case Prediction	81.58%	97.78%
Loss Prediction(observation point)	83.63%	98.1%
Loss Prediction(bankruptcy point)	81.6%	97.12%

Table 5.8: Prediction performance of our system

5.7 Chapter summary

Personal bankruptcy prediction is a challenging application in the financial industry. Credit scoring models are commonly used in this important application domain.

[9] Free Credit Reports in Canada. Available: `https://www.freecreditreportsincanada.ca`

Some data mining models have also been investigated in this area. Both credit scoring models and existing data mining models ignore the sequence pattern of clients' behavior, which has proven to be very capable of identifying bad accounts in our investigation.

We have investigated the use of categorical data mining and sequence mining techniques described in Chapter 2 and Chapter 4 in personal bankruptcy prediction. Some discriminative patterns are discovered by our clustering techniques, which are referred to as bankruptcy features. The bankruptcy features used as predictors to identify bad accounts are readily comprehensible, making our prediction system easy to explain to creditors. The experimental results show that the bankruptcy features have a great capacity to identify bad accounts, especially in terms of loss prediction. Our prediction system based on bankruptcy features outperforms the credit bureau credit scoring method despite the fact that it uses a single data source, whereas the scoring system is run based on multiple data sources relating to creditworthiness. Our final system based on combining bankruptcy features and some existing features demonstrates promising prediction performance. Our system can be used as an important supplement to existing credit scoring systems, which have an intrinsic deficiency when it comes to loss prediction.

Chapter 6

Conclusions and Future Work

In this thesis, the focus of our research was on clustering categorical and sequence data, and their application in the domain of personal bankruptcy prediction. Clustering categorical and sequence data is much more challenging than clustering numeric data due to the lack of inherently meaningful similarity measure. In our proposed algorithms, i.e., DHCC, SDHCC, model-based K-means, we circumvented the obstacle of defining the naturally meaningful pairwise similarity, by defining the similarities between individual object and a cluster of objects. Their performance and usefulness were extensively validated by experiments on both public data and the data from our project on a major Canadian bank.

We proposed a novel divisive hierarchical clustering algorithm for categorical data, named DHCC. In DHCC, we set the objective of clustering categorical data to minimizing the objective function, i.e., the sum of Chi-square error (SCE) and proposed a systemic approach to optimize the objective function. Each bisection step of DHCC consists of two phases: preliminary bisection based on multiple correspondence analysis (MCA) and iterative refinement based Chi-square distance measure between a single categorical object and a cluster of categorical objects. MCA plays an important role in bisection by taking the global information on the data distribution into account, and furthermore, the preliminary bisection based on MCA is closely related to optimization of the objective function. The refinement phase further (locally) optimizes the objective function by relocating the objects from the cluster being split.

Using indicator matrix to represent the categorical data and Chi-square distance to measure the dissimilarity between an individual object and a cluster makes DHCC capable of seamlessly discovering clusters embedded in subspace.

We proposed a semi-supervised divisive hierarchical clustering algorithm for categorical data, named SDHCC, which is underlain by DHCC. We investigated how to take advantage of prior background knowledge, which is provided as *must-link* and *cannot-link* constraints on pairs of instances, in clustering categorical data. In SDHCC, we view semi-supervised clustering of categorical data as an issue of optimizing the objective function SCE subject to extra instance-level constraints, and proposed a systematic approach to deal with this problem. The optimization process in SDHCC consists of three phases: initialization of bisection; iterative refinement of the bisection based on Chi-square distance; and alleviation of the *cannot-link* violation. These three steps guide the optimization process in SDHCC to a better solution in terms of satisfying the constraints, leading to remarkable improvement over the unsupervised clustering algorithm DHCC.

We presented a statistical model, i.e., conditional probability distribution (CPD), for sequence clustering. Based on the CPD model, a model-based K-means algorithm was proposed for clustering categorical sequences. CPD model is among the short memory approaches, which are based on the hypothesis that the probability distribution of the next symbol given the preceding segment can be approximated by observing the last L symbols in that segment. Compared with Hidden Markov Models with fixed memory length L, CPD model has varying L depending on the statistical structure of the sequence cluster, which is more adaptive to the real complex situation where the significant features of the sequences exist in different length. The model-base K-means algorithm works in a similar way to the traditional K-means on vectorial data, and suitable for both long and short categorical sequences.

On the project of personal bankruptcy prediction, we explored the extent to which the use of clustering techniques proposed in this thesis can discover useful and comprehensible discriminative features, which can help to solve the problem of personal bankruptcy prediction. We generated two types of data, i.e., categorical data and behavior sequences, from the preprocessing step on the high-dimensional and complex

original data. Then we applied our DHCC and model-based K-means algorithms on these two types of data respectively, and discover some predictive features. These mined bankruptcy features, combined with some existing prediction-capable features, are represented in low-dimensional Boolean vector space, and SVM was exploited to combine these features and generate a confidence score for each account. Our final prediction system demonstrated promising prediction performance, especially in loss prediction. Our research also demonstrated that the sequence pattern of clients' behavior is very capable of identifying bad accounts in personal bankruptcy prediction.

Overall, the research presented in this thesis has made significant contributions in the domain of unsupervised and semi-supervised clustering of categorical data, clustering sequence data, as well as personal bankruptcy prediction. The algorithms proposed in this thesis can be used as important tools for exploratory data analysis. Our investigation in this thesis also bridges the industry-academia gap in the domain of personal bankruptcy prediction, and opens another avenue for the future research in this domain, as well as the similar application fields of classification and prediction.

Our research also inspires some future work. We present some as follows:

- Anomalous objects (also referred to as outliers) in the data can affect DHCC. These objects can distort the definition of the cluster center to some extent, and in rare cases, make the refinement process fail to converge. Removing anomalous records prior to clustering can improve clustering performance. An interesting area for future research would thus be to design a seamless strategy or algorithm to detect the outliers for DHCC. In fact, outlier detection from categorical data is a very important and largely open question [21].

- In semi-supervised clustering with instance-level constraints, some constraints may be mis-specified by the user. For example, in the case that some instances having the same class label but should actually be placed in different clusters in a proper clustering, the domain expert may assume they are in the same cluster; and also in real-life data, it commonly happens that some very similar, even identical, instances actually belong to different classes. The *must-link* and *cannot-link* constraints on these instances mislead the clustering process. How

to extent SDHCC to tolerate the mis-specified constraints to make the semi-supervised clustering algorithm more robust is interesting for future research.

- The model-based K-means for clustering sequence proposed in this thesis is very effective in discovering the discriminative sequence patterns in the application of personal bankruptcy prediction. However, it has some drawbacks, such as it needs the assumption of the number of clusters, and is not very stable due to random model initialization. Designing a divisive hierarchical clustering algorithm for sequences, which is based on CPD model, and parameter-free, is challenging yet meaningful subject for future work.

Bibliography

[1] ABDI, H., AND VALENTIN, D. Multiple correspondence analysis. In *Encyclopedia of Measurement and Statistics*, N. Saltkind, Ed. SAGE, 2007.

[2] AGGARWAL, C. On the use of wavelet decomposition for string classification. *Data Mining and Knowledge Discovery 10* (2005).

[3] AGGARWAL, C. C., WOLF, J. L., YU, P. S., PROCOPIUC, C., AND PARK, J. S. Fast algorithms for projected clustering. In *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data* (1999).

[4] ALOISE, D., DESHPANDE, A., HANSEN, P., AND POPAT, P. Np-hardness of euclidean sum-of-squares clustering. *Machine learning 75* (2009).

[5] ANDRITSOS, P., TSAPARAS, P., MILLER, R., AND SEVCIK, K. LIMBO: Scalable clustering of categorical data. *Lecture Notes in Computer Science* (2004).

[6] BAR-HILLEL, A., HERTZ, T., SHENTAL, N., AND WEINSHALL, D. Learning distance functions using equivalence relations. In *Proceedings of the 20th International Conference on Machine Learning* (2003).

[7] BARBARA, D., LI, Y., AND COUTO, J. COOLCAT: An entropy-based algorithm for categorical clustering. In *Proceedings of the 11th ACM International Conference on Information and Knowledge Management* (2002).

[8] BASU, S., BANERJEE, A., AND MOONEY, R. Semi-supervised clustering by seeding. In *Proceedings of the 19th International Conference on Machine Learning* (2002).

[9] BASU, S., BILENKO, M., AND MOONEY, R. A probabilistic framework for semi-supervised clustering. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2004).

[10] BEJERANO, G., AND YONA, G. Modeling protein families using probabilistic suffix trees. In *Proceedings of the 3th Annual International Conference on Research in Computational Molecular Biology* (1999).

[11] BILENKO, M., BASU, S., AND MOONEY, R. Integrating constraints and metric learning in semi-supervised clustering. In *Proceedings of the 21th International Conference on Machine Learning* (2004).

[12] BILENKO, M., AND MOONEY, R. Adaptive duplicate detection using learnable string similarity measure. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2003).

[13] BOUGUESSA, M., AND WANG, S. Mining projected clusters in high-dimensional spaces. *IEEE Transactions on Knowledge and Data Engineering 21* (2009).

[14] BRAND, M. Fast low-rank modifications of the thin singular value decomposition. *Linear Algebra and its Applications* (2006).

[15] CESARIO, E., MANCO, G., AND ORTALE, R. Top-down parameter-free clustering of high-dimensional categorical data. *IEEE Transactions on Knowledge and Data Engineering 19* (2007).

[16] CHAPPELLE, O., SCHÖLKOPF, B., AND ZIEN, A. *Semi-Supervised Learning.* MIT Press, Cambridge,MA, 2006.

[17] CHEN, H.-L., CHUANG, K.-T., AND CHEN, M.-S. On data labeling for clustering categorical data. *IEEE Transactions on Knowledge and Data Engineering 20* (2008), 1458–1472.

[18] CHEN, K., AND LIU, L. The 'best k' for entropy-based categorical data clustering. In *Proceedings of the 17th International Conference on Scientific and Statistical Database Management* (2005).

[19] CORMEN, T., LEISERSON, C., AND RIVEST, R. *Introduction to algorithms.* MIT Press, Cambridge,MA, 1990.

[20] CRISTIANINI, N., AND SHAWE-TAYLOR, J. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods.* Cambridge University Press, 2000.

[21] DAS, K., AND SCHNEIDER, J. Detecting anomalous records in categorical datasets. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2007).

[22] DAVIDSON, I., AND RAVI, S. Clustering with constraints: Feasibility issues and the k-means algorithm. In *Proceedings of the 5th SIAM International Conference on Data Mining* (2005).

[23] DAVIDSON, I., AND RAVI, S. Using instance-level constraints in agglomerative hierarchical clustering: theoretical and empirical results. *Data mining and Knowledge Discovery 18* (2009).

[24] DESHPANDE, M., AND KARYPIS, G. Evaluation of techniques for classifying biological sequences. Tech. rep., University of Minnesota, 2001.

[25] DHILLON, I. S., AND MODHA, D. S. Concept decompositions for large sparse text data using clustering. *Machine Learning 42* (2001).

[26] DING, C., AND HE, X. Cluster merging and splitting in hierarchical clustering algorithms. In *Proceedings of the 2nd IEEE International Conference on Data Mining* (2002).

[27] DO, H., AND KIM, J. Categorical data clustering using the combinations of attribute values. *Lecture Notes in Computer Science* (2008).

[28] DONATO, J., SCHRYVER, J., AND HINKEL, G. Mining multidimensional data for decision support. *Future generation computer systems 15* (1999).

[29] DRINEAS, P., DRINEA, E., AND HUGGINS, P. An experimental evaluation of a monte-carlo algorithm for singular value decomposition. *Lecture Notes in Computer Science* (2003).

[30] DUDA, R., HART, P., AND STORK, D. *Pattern Classification*, second ed. Wiley, New York, 2001.

[31] ESTER, M., KRIEGEL, H., SANDER, J., AND XU, X. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 2nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (1996).

[32] EVERITT, B., LANDAU, S., AND LEESE, M. *Cluster Analysis*, fourth ed. Arnold Publishers, London, 2001.

[33] FISHER, D. Iterative optimization and simplification of hierarchical clustering. *Journal of Artificial Intelligence Research 4* (1996).

[34] GAN, G., AND WU, J. Subspace clustering for high dimensional categorical data. *SIGKDD Explorations 6*, 2 (2004).

[35] GANTI, V., GEHRKE, J., AND RAMAKRISHNAN, R. CACTUS: Clustering categorical data using summaries. In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (1999).

[36] GOMORY, R., AND HU, T. Multi-terminal network flows. *Journal of the Society for Industrial and Applied Mathematics 9*, 4 (1961), 551–570.

[37] GRABMEIER, J., AND RUDOLPH, A. Techniques of cluster algorithms in data mining. *Data Mining and Knowledge Discovery 6*, 4 (2002).

[38] GREENACRE, M. *Correspondence Analysis in Practice.* Academic Press, London, 1993.

[39] GREENACRE, M., AND BLASIUS, J. *Multiple Correspondence Analysis and Related Methods.* Chapman & Hall, 2006.

[40] GUHA, S., RASTOGI, R., AND SHIM, K. ROCK: A robust clustering algorithm for categorical attributes. In *Proceedings of the 15th IEEE International Conference on Data Engineering* (1999).

[41] HARTIGAN, J., AND WONG, M. A k-means clustering algorithm. *Applied statistics 28* (1979).

[42] HE, J., SHI, Y., AND XU, W. Classifications of credit card holder behavior by using multiple criteria nonlinear programming. In *Proceedings of Chinese Academy of Sciences Symposium on Data Mining and Knowledge Management* (2004).

[43] HÖPPNER, F., KLAWONN, F., KRUSE, R., AND RUNKLER, T. *Fuzzy Cluster Analysis: Methods for Classification, Data Analysis and Image Recognition.* Wiley, New York, 1999.

[44] HUANG, J. Z., NG, M. K., RONG, H., AND LI, Z. Automated variable weighting in k-means type clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence 27*, 5 (2005), 657–668.

[45] HUANG, Z. Extensions to the k-means algorithm for clustering large data sets with categorical value. *Data Mining and Knowledge Discovery 2* (1998), 283–304.

[46] JAIN, A., MURTY, M., AND FLYNN, P. Data clustering: A review. *ACM Computing Surveys 31*, 3 (1999).

[47] JOLSON, B. Predicting insolvency and preventing losses using transaction analysis. Tech. rep., Fair Isaac Corporation, 2007.

[48] KARYPIS, G., HAN, E., AND KUMAR, V. Multilevel refinement for hierarchical clustering. Tech. rep., University of Minnesota, 1999.

[49] KELIL, A., AND WANG, S. SCS: A new similarity measure for categorical sequences. In *Proceedings of the 8th IEEE International Conference on Data Mining* (2008).

[50] KEOGH, E., LONARDI, S., AND RATANAMAHATANA, C. Towards parameter-free data mining. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2004).

[51] KLEIN, D., KAMVAR, S., AND MANNING, C. From instance-level constraints to space-level constraints: making the most of prior knowledge in data clustering. In *Proceedings of the 19th International Conference on Machine Learning* (2002).

[52] KOMORAD, K. On credit scoring estimation. Master's thesis, Humboldt University, Germany, 2002.

[53] KONDRAK, G. N-gram similarity and distance. In *Proceedings of the 12th International Conference on String Processing and Information Retrieval* (2005).

[54] KULIS, B., BASU, S., DHILLON, I., AND MOONEY, R. Semi-supervised graph clustering: a kernel approach. *Machine Learning 74* (2009), 1–22.

[55] KUMAR, N., AND KUMMAMURU, K. Semisupervised clustering with metric learning using relative comparisons. *IEEE Transactions on Knowledge and Data Engineering 20* (2008), 496–503.

[56] LELIS, L., AND SANDER, J. Semi-supervised density-based clustering. In *Proceedings of the 9th IEEE International Conference on Data Mining* (2009).

[57] LI, T., MA, S., AND OGIHARA, M. Entropy-based criterion in categorical clustering. In *Proceedings of the 21st International Conference on Machine Learning* (2004).

[58] LOPRESTLI, D., AND TOMKINS, A. Block edit models for approximate string matching. *Theoretical Computer Science 181* (1997).

[59] LU, Y., WANG, S., LI, S., AND ZHOU, C. Particle swarm optimizer for variable weighting in clustering high-dimensional data. *Machine Learning 82* (2011).

[60] M., F., P., F., AND S., M. Overcoming the memory bottleneck in suffix tree construction. In *Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science* (1998).

[61] MAYS, E. *Credit Scoring for Risk Managers: The Handbook for Lenders.* Thomson, Mason, U.S., 2005.

[62] MESSAOUD, R., BOUSSAID, O., AND RABASEDA, S. Efficient multidimensional data representations based on multiple correspondence analysis. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2006).

[63] MUTHUKRISHNAN, S., AND SAHINALP, S. Approximate nearest neighbors and sequence comparison with block operations. In *Proceedings of ACM symposium on Theory of computing* (2000).

[64] NG, R. T., AND HAN, J. Efficient and effective clustering methods for spatial data mining. In *Proceedings of the 20th Conference on VLDB* (1994).

[65] NIGAM, K. *Using unlabeled data to improve text classification.* PhD thesis, Carnegie Mellon University, 2001.

[66] PARSONS, L., HAQUE, E., AND LIU, H. Subspace clustering for high dimensional data: A review. *ACM SIGKDD Explorations Newsletter 6*, 1 (2004).

[67] PENG, Y., KOU, G., SHI, Y., AND CHEN, Z. Improving clustering analysis for credit card accounts classification. *Lecture Notes in Computer Science 3516* (2005), 548–553.

[68] QUINLAN, J. *C4.5: Programs for machine learning.* Morgan Kaufmann, San Fransisco, U.S., 1993.

[69] RON, D., SINGER, Y., AND TISHBY, N. The power of amnesia: learning probabilistic automata with variable memory length. *Machine Learning 25* (1996), 117–149.

[70] RUIZ, C., SPILIOPOULOU, M., AND MENASALVAS, E. Density-based semi-supervised clustering. *Data Mining and Knowledge Discovery 21* (2010), 345–370.

[71] SALVADOR, S., AND CHAN, P. Determining the number of clusters/segments in hierarchical clustering/segmentation algorithms. In *Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence* (2004).

[72] SAN, O., HUYNH, V., AND NAKAMORI, Y. An alternative extension of the k-means algorithm for clustering categorical data. *International Journal of Applied Mathematics and Computer Science 14*, 2 (2004).

[73] SHENTAL, N., BAR-HILLEL, A., HERTZ, T., AND WEINSHALL, D. Computing gaussian mixture models with em using equivalence constraints. In *In Advances in Neural Information Processing Systems 16* (2003), MIT Press.

[74] SMYTH, P. clustering sequences with hidden markov models. In *Advances in neural information processing system* (1997).

[75] SUGAR, C., AND JAMES, G. Finding the number of clusters in a dataset. *Journal of the American Statistical Association 98*, 463 (2003), 750–763.

[76] SUN, H., WANG, S., AND JIANG, Q. FCM-based model selection algorithm for determining the number of clusters. *Pattern Recognition 37*, 10 (2004).

[77] TAN, P., STEINBACH, M., AND KUMAR, V. *Introduction to Data Mining.* Addison-Wesley, 2005.

[78] TANG, W., XIONG, H., ZHONG, S., AND WU, J. Enhancing semi-supervised clustering: A feature projection perspective. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2007).

[79] THOMAS, L., EDELMAN, D., AND CROOK, J. *Credit Scoring and Its Applications.* SIAM, 2002.

[80] TSUKIMOTO, H. Logical regression analysis: from mathematical formulas to linguistic rules. In *In Foundations and advances in data mining*, W. Chu and T. Lin, Eds., vol. I80. Springer, 2005.

[81] Ukkonen, E. On-line construction of suffix trees. *Algorithmica 14* (1995), 249–260.

[82] Wagstaff, K., Cardie, C., Rogers, S., and Schroedl, S. Constrained k-means clustering with background knowledge. In *Proceedings of the 18th International Conference on Machine Learning* (2001).

[83] Wang, K., Xu, C., and Liu, B. Clustering transactions using large items. In *Proceeding of the 10th ACM conference on Information and knowledge Management(CIKM)* (1999).

[84] Wu, J., Xiong, H., and Chen, J. Adapting the right measures for k-means clustering. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2009).

[85] Wu, X., Kumar, V., Quinlan, J. R., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G., Ng, A., Liu, B., Yu, P., Zhou, Z.-H., Steinbach, M., Hand, D., and Steinberg, D. Top 10 algorithms in data mining. *Knowledge and Information System 14* (2008).

[86] Xia, F., Zhang, W., Li, F., and Yang, Y. Top 10 algorithms in data mining. *Knowledge and Information System 17* (2008), 381–395.

[87] Xing, E., Ng, A., Jordan, M., and Russell, S. Distance metric learning with application to clustering with side-information. In *Advance in Neural Information Processing System*, vol. 15. MIT Press, Cambridge, MA, 2003.

[88] Xiong, T., Wang, S., Mayers, A., and Monga, E. Personal bankruptcy prediction using sequence mining. In *Proceeding of KDD2008 Workshop on Data Mining for Business Applications* (2008).

[89] Xiong, T., Wang, S., Mayers, A., and Monga, E. A new MCA-based divisive hierarchical algorithm for clustering categorical data. In *Proceedings of the 9th IEEE International Conference on Data Mining(ICDM)* (2009).

[90] XIONG, T., WANG, S., MAYERS, A., AND MONGA, E. DHCC: Divisive hierarchical clustering of categorical data. Data Mining and Knowledge Discovery (accepted), 2011.

[91] XIONG, T., WANG, S., MAYERS, A., AND MONGA, E. Semi-supervised parameter-free divisive hierarchical clustering of categorical data. Proceedings of the 15th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD2011, in press), 2011.

[92] YAN, H., CHEN, K., AND LIU, L. Efficiently clustering transactional data with weighted coverage density. In *Proceeding of the 17th ACM conference on Information and knowledge Management(CIKM)* (2006).

[93] YANG, J., AND WANG, W. CLUSEQ: efficient and effective sequence clustering. In *Proceedings of the 19th IEEE International Conference on Data Engineering (ICDE)* (2003).

[94] YANG, Y., GUAN, X., AND YOU, J. CLOPE: A fast and effective clustering algorithm for transactional data. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2002).

[95] ZHAO, Y., AND KARYPIS, G. Hierarchical clustering algorithms for document datasets. *Data Mining and Knowledge Discovery 10* (2005).

[96] ZHU, X. Semi-supervised learning literature survey. `http://pages.cs.wisc.edu/~jerryzhu/pub/ssl_survey.pdf`, 2008.